Received 28 December 2022, accepted 16 January 2023, date of publication 18 January 2023, date of current version 24 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3237971

RESEARCH ARTICLE

An IoT Toolchain Architecture for Planning, Running and Managing a Complete Condition Monitoring Scenario

FEDERICO MONTORI[1,2], (Member, IEEE), IVAN ZYRIANOFF[1,2], LORENZO GIGLI[1,2], (Graduate Student Member, IEEE), ALESSANDRO CALVIO[1], RICCARDO VENANZI[1], (Member, IEEE), SIMONE SINDACO[2], LUCA SCIULLO[1], (Member, IEEE), FEDERICA ZONZINI[2,3], (Member, IEEE), MATTEO ZAULI[2,3], (Member, IEEE), NICOLA TESTONI[2,3], (Member, IEEE), ALESSANDRO BERTACCHINI[4], (Member, IEEE), ELISA LONDERO[5], ENRICO ALESSI[6], MARCO DI FELICE[1,2], (Member, IEEE), LUCIANO BONONI[1], PAOLO BELLAVISTA[1], (Senior Member, IEEE), LUCA DE MARCHI[2,3], (Senior Member, IEEE), ALESSANDRO MARZANI[2,7], PAOLO AZZONI[5], AND TULLIO SALMON CINOTTI[2]

[1]Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy
[2]Advanced Research Center on Electronic Systems "Ercole De Castro", University of Bologna, 40126 Bologna, Italy
[3]Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi", University of Bologna, 40126 Bologna, Italy
[4]Department of Science and Engineering Methods, University of Modena and Reggio Emilia, 41121 Modena, Italy
[5]Eurotech S.p.A., 33020 Udine, Italy
[6]Advanced Research and System Platforms RND, Analog, MEMS and Sensors Group, STMicroelectronics, 95121 Catania, Italy
[7]Department of Civil, Chemical, Environmental, and Materials Engineering, University of Bologna, 41121 Modena, Italy

Corresponding author: Federico Montori (federico.montori2@unibo.it)

This work was supported by the Electronic Components and Systems for European Leadership, ECSEL, Joint Undertaking under Grant 826452; in part by the Arrowhead Tools European Union Horizon 2020 Research and Innovation Program; in part by the member states, and by Istituto nazionale Assicurazione Infortuni sul Lavoro (INAIL) under Grant BRIC/2018; and in part by the ID=11 Framework, under Project MAC4PRO, (Smart maintenance of industrial plants and civil structures via innovative monitoring technologies and prognostic approaches).

ABSTRACT Condition Monitoring (CM) is an extremely critical application of the Internet of Things (IoT) within Industry 4.0 and Smart City scenarios, especially following the recent energy crisis. CM aims to monitor the status of a physical appliance *over time* and in *real time* in order to react promptly when anomalies are detected, as well as perform predictive maintenance tasks. Current deployments suffer from both interoperability and management issues within their engineering process at all phases – from their design to their deployment, to their management –, often requiring human intervention. Furthermore, the fragmentation of the IoT landscape and the heterogeneity of IoT solutions hinder a seamless onboarding process of legacy devices and systems. In this paper, we tackle these problems by first proposing an architecture for CM based on both abstraction layers and toolchains, *i.e.*, automated pipelines of engineering tools aimed at supporting the engineering process. In particular, we introduce four different toolchains, each of them dedicated to a well-defined task (*e.g.*, energy monitoring). This orthogonal separation of concerns aims to simplify both the understanding of a complex ecosystem and the accomplishment of independent tasks. We then illustrate our implementation of a complete CM system that follows said architecture as a real Structural Health Monitoring (SHM) pilot of the Arrowhead Tools project, by describing in detail every single tool that we developed. We finally show how our pilot achieves the main objectives of the project: the reduction of engineering costs, the integration of legacy systems, and the interoperability with IoT frameworks.

INDEX TERMS IoT, WoT, condition monitoring, SHM, arrowhead, toolchain.

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Tucci.

I. INTRODUCTION

The Internet of Things (IoT) is now a dominant technology for building automation and industrial solutions and the basis for a variety of application domains [1]. One such is Condition Monitoring (CM), which is crucial for several scenarios in Industry 4.0. CM is responsible for monitoring a defined physical appliance – or a set of those – via a number of sensors that are deployed in the field. The goal is the monitoring of certain condition parameters, such as vibrations in machinery, in order to identify abnormal values and act in advance in case a possible fault is detected or predicted to occur (as in predictive maintenance) [2].

The implementation of CM is widespread in several use cases, for instance, wind turbines or railway tracks [3], however, the lack of proper automation over its engineering process may easily lead to overwhelming deployment and operation costs. Indeed, managing remotely and automatically CM scenarios and, in general, promoting the automation of processes, is a very crucial aspect in order to avoid significant manual work [4]. This overhead is the major cause of heavy engineering costs and the slowdown of processes. Another major cause is the fragmentation of IoT technologies, which leads to severe interoperability issues that make the integration of legacy technologies – always present in industrial monitoring scenarios – challenging and costly. In certain cases this problem is mitigated by established IoT frameworks, however, the transition to a single technology is rarely free of technological obstacles.

In order to tackle these problems, in this paper we design and propose a toolchain architecture for CM of industrial scenarios that are characterized by data coming from a set of heterogeneous sensors. The architecture leverages indeed the concept of "toolchain", a set of software engineering tools that are connected to each other via interfaces, thus enabling automation across abstraction layers. In our case, the toolchain architecture aims to support a defined engineering process following the principles of the EU Arrowhead Tools project,[1] in order to meet certain project objectives. In particular, in this paper we describe four different and generalizable toolchains, altogether concurring in achieving the following specific goals: (i) integrating third-party and legacy devices through a seamless onboarding procedure supported by translation of standards, (ii) automatically adjusting the working conditions of the devices in order to optimize their maintenance and management costs according to the conditions of the environment, and (iii) supporting the integration with well-established IoT frameworks and cloud services to lower the need for on-site intervention. Within our proposal, the main enabler of toolchains is the Eclipse Arrowhead framework, which provides local clouds with service-oriented capabilities, such as loose coupling, discovery, and orchestration, and represents a single fruition channel for service providers and consumers.

[1]https://tools.arrowhead.eu/home/

As a proof-of-concept implementation, we then discuss a use-case related to Structural Health Monitoring (SHM), which is also a demonstrator for the Arrowhead Tools project, also available in a video.[2] SHM, a particular instance of CM, is a relatively young discipline aimed at assessing the integrity condition of existing structures and critical assets by means of on-condition inspection procedures, meaning that the target structure can be controlled in real-time and over time [5], [6]. SHM shows two main benefits over standard, scheduled maintenance policies: first of all, it prevents the normal serviceability to be interrupted, since the target structure can be inspected while in its operative conditions, and, secondly, it assists inspectors in the decision-making process via automated assessment procedures. However, despite the significant advancements achieved by the information and structural engineering community, the fulfillment of these functionalities compatible with the need for a safer, green, and more digital industry still poses major concerns that demand novel maintenance strategies and architectures, such as in our proposal [7]. The current baseline is an SHM sensor network, hosting a number of inertial sensors that monitor the vibrations of a model building. We firstly provided sensors with a degree of interoperability by leveraging the W3C Web of Things (WoT) standard [8]. The W3C WoT is a recent standardization effort that abstracts simple sensors and actuators into Web resources that bear a detailed description of their capabilities, their interaction affordances, and their semantics [9]. In our case, each of the SHM sensors is equipped with a WoT counterpart that governs all the incoming and outgoing communication. The baseline is enriched by the development of a number of engineering tools, all of them described in detail, that comply with our proposed toolchain architecture, making the use case an instance of our proposal.

We then validate the advantages of our approach through selected experiments and we report the observed system behavior utilizing the MODRON [10] platform. The paper is structured as follows: Section II introduces the Arrowhead Tools project and its related definitions, as well as the Eclipse Arrowhead framework. Section III presents our proposed toolchain architecture and how our toolchains are aiming at fulfilling the project objectives. Section IV presents our demonstrator pilot, the sensor used, and the tools for their interoperability. Section V details the implemented tools and components for the four proposed toolchains within our demonstrative implementation. Section VI describes the experiments and the validation results with a particular focus on the project objectives and discusses takeaway messages. Section VII discusses related works to better frame our proposal within the literature. Finally, Section VIII concludes the paper and discusses some future directions of research.

II. THE ARROWHEAD ECOSYSTEM

The use case described in this paper has been developed within the scope of the ECSEL Arrowhead Tools project,

[2]https://youtu.be/1nEOJpXu9l8

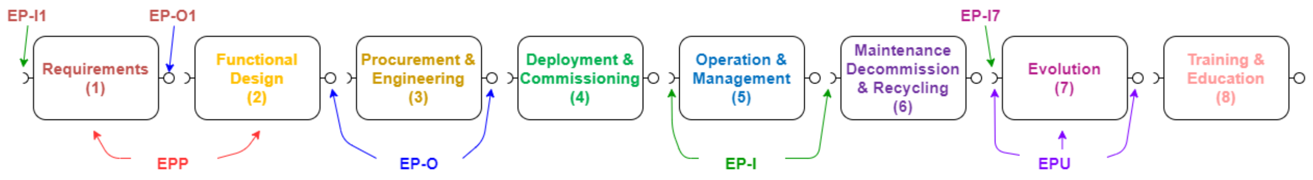

EP-I1
EP-O1
EP-I7
Requirements (1)
Functional Design (2)
Procurement & Engineering (3)
Deployment & Commissioning (4)
Operation & Management (5)
Maintenance Decommission & Recycling (6)
Evolution (7)
Training & Education (8)
EPP
EP-O
EP-I
EPU

FIGURE 1. **A graphical overview of the Arrowhead Tools Engineering Process.**

as one of the main demonstrators. This section aims to give an overview of the project and a handful of general definitions necessary to understand the concepts of engineering process, tool, toolchain, and core services.

A. THE ARROWHEAD TOOLS PROJECT

The ECSEL Arrowhead Tools project (running from 2019 to 2022) is one of the biggest and most influential EU projects of the latest years. It features about 90 partners from 17 EU countries and has triggered an investment of more than 100 million Euros, shared between the industrial partners, the countries, and the EU commission.

The project aims to provide automation and digitalization technologies and solutions to the European industry in order to address the integration gaps of information and operation technologies. This is achieved by the development of new open-source tools that operate at design time and run time in the engineering process of IoT industrial artifacts.

Within the Arrowhead Tools project, a detailed *engineering process model* has been proposed lately in [11] and [12] and to which we will refer to as Arrowhead Tools Engineering Process (AHT-EP). The current version of the AHT-EP features concepts that are compatible with earlier similar models, in order to maintain backward compliance and at the same time retain the flexibility to meet current industrial requirements. The AHT-EP is built on top of the well-known *Extended Automation Engineering Model* defined in the ISO/IEC 81346 standard [13]. However, it allows a decoupled flow of information through the engineering process phases (EPPs) which no longer need to be traversed in a fixed order, if needed. The eight EPPs are depicted in Figure 1 and are connected via interfaces. Each EPP has an *incoming* (EP-I) and an *outgoing* (EP-O) interface. The term engineering process unit (EPU) means any of the three concepts. Now, the main activities within the Arrowhead Tools project demand the achievement of the project objectives via the development of software *tools*. Tools in the project have been precisely defined as software products (or hardware with adequate software on board) that improve an already established industrial baseline by supporting one or more EPPs [14]. The complete definition features additional properties, such as atomicity and interoperability, which enable the combination of a set of tools, applied in sequence, that can supervise the whole engineering process of a certain artifact. This makes tools suitable for compositional architectures, called *toolchains*, which are simply defined as the sequential composition of engineering tools within a fully fledged engineering process. Finally, tools and toolchains are designed to fulfill a set of project objectives that provide defined KPIs for the evaluation of the impact of the tool on the baseline:

1) The seamless integration of legacy components into the architecture.
2) The reduction of engineering costs.
3) The interoperability with established IoT frameworks.

B. THE ECLIPSE ARROWHEAD FRAMEWORK

The last decade has been dominated by a fast-paced industrial revolution, particularly affecting IoT-based ecosystems. In particular, Industry 4.0 no longer relies on legacy and monolithic SCADA/DCS systems, instead, they are supported by flexible Service-Oriented Architectures (SOA), where modular systems consume or provide services, ensuring loose coupling between modules and their reusability across multiple domains [15]. The Eclipse Arrowhead Framework is a software platform released as an open source product of the Arrowhead Project[3] that structures closed environments as Local Clouds: controlled ecosystems that implement the base concepts of SOA – loose coupling, late binding, and discovery – and hosts a single instance of a central coordination entity, called the Core Services [16]. Then, each Local Cloud acts as a System-of-Systems in which each system is either an *application system* (if an integral part of the baseline) or a *tool* (if it meets our previous definition). Regardless, they behave as service providers (servers exposed through endpoints) and/or service consumers (clients that query other services). Service consumption is supervised and managed by the Core Services, which must be deployed in the Local Cloud in a minimum set. The latter defines the ''mandatory'' Core Services in order to be Arrowhead-compliant; which are the Service Registry, the Authorization, and the Orchestration. The **Service Registry** retains a service record – a set of metadata – for each of the services in the Local Cloud acting as a registrar which enables discovery and loose coupling. Service providers can register themselves or a modeling facility may be used in the design phase of the Local Cloud, such as SysML [17]. The **Authorization** is a storage of a set of authorization rules that specify whether a consumer is authorized to use a certain service. In addition, it provides a token-based authentication mechanism. Finally, the **Orchestration** is the enabler of late

[3]http://www.arrowheadproject.eu/

binding, as it allows an additional actor, the cloud manager, to associate directly consumers to providers at run-time. This way, consumers cannot autonomously decide which service provider to query, instead, they query the Orchestration service to obtain the provider that was assigned to them.

III. TOOLCHAIN ARCHITECTURE

In this section, we present the overall architecture of a toolchain-enabled complex condition monitoring scenario. We extend the 4-layer architecture build for SHM (see Section I) initially proposed in [10], and we take inspiration from the 5-layer IoTecture defined in [18]. The final objective is to present a high-level generic and flexible architecture capable of supporting deployments comprised of numerous heterogeneous devices, different communication technologies, applications with non-uniform interfaces, and multiple end-user roles, ranging from managers to data scientists and system administrators.

Figure 2 depicts the layered architecture adopted. Each layer defines a specific function, which can be performed by one or multiple applications, that connects with the other adjacent layers.

The **L1 - Physical** layer encompasses devices responsible for interacting with the physical world. This category comprises all sensors and actuators, such as accelerometers, gas, and piezoelectric sensors. This layer also includes the physical communication medium of the network stack – *e.g.*, Wi-Fi, LTE, LoRa, etc.

The **L2 - Interoperability** was formally referred to as data acquisition [10] or as communication layer [18]. We expanded the responsibilities of this layer since it enables seamless communication between the lower and upper layers. Hence, this category is comprised of two sets of tools:

1) Communication-enablers: the tools responsible for data to be delivered, such as protocol-specific message brokers (*e.g.*, MQTT Brokers) and network layer enablers, (*e.g.*, the LoRaWAN server stack);
2) Homogenization tools: provide uniform interfaces to consume data (*e.g.*, the WoT standard) and bridge different data structures or protocols.

The **L3 - Operation** layer encompasses tools that are responsible for the data storing, filtering, processing, and transformation operations. This includes time-series databases and supporting applications that manage the data insertion and acquisition operations, optimization algorithms that either aid users in making informed decisions upon the system or act autonomously, Digital Twins that predict the future state of the system. Data transformation tools are also part of this layer since sensor data generally needs calibration or some data processing – *e.g.*, the cumulative event sum is a crucial metric in the field of SHM (an event is registered if a sensor is excited beyond a certain threshold); to calculate it we need not only the definition of a threshold, but the history of the system through time.

The **L4 - Service** layer is comprised of services for end-users, responsible for supporting their interaction with the system. This includes data visualization as well as interactions that can potentially lead up to commands that change the current state of the system. The visualization tools should consider that complex systems are managed by users performing different roles, each with their concerns and experiences. For instance, a dashboard that enables civil engineers to assess the health of a given structure is different from a GUI that assists the system administrator with the current state of each software application.

On the right-hand side of Figure 3 there is a security block that crosses all layers. The reason is that each layer needs to address its security liabilities since they impose different requirements and constraints. Hence, end-user authorization and authentication – performed in L4-Service – needs to be handled differently as the same operations for end-devices – managed by L2-Interoperability.

The building blocks for IoT data acquisition and processing are already a well-established pipeline in literature [19]. Although adequate for most scenarios, CM has multiple moving pieces, since it requires scalability and it is an inherently interdisciplinary domain. The IoT pipeline is the backbone of the system. However, when the system scales up, it requires more effort and costs to efficiently manage applications, devices, and energy. In Section II-A we set three project objectives for a CM scenario, in order to satisfy a number of KPIs that are necessary for the current standards. In order to fulfill said objectives, we propose four different toolchains – as illustrated in Figure 2 –, each one enabling a distinct functionality and composing software tools, that perform tasks determined by their own layer, into pipelines.

The first layer is shared among the toolchains since the physical components are the base of any IoT-based system. The Interoperability layer operates as a toolbox, providing general-purpose tools to each of the toolchains, each of which selects a subset of components in order to enable its own interoperability. Tools from L2 up to L4 may be deployed in the cloud, or in a local cloud (*i.e.*, locally with the structure). Interactions among tools and application systems in a local cloud are mediated by a local instance of the Eclipse Arrowhead framework, which also holds a connection with the outer world (*i.e.*, the cloud). Following is the description of our proposed four toolchains:

- **Data Toolchain**: it is the traditional IoT pipeline, enabling the main goal of the system. In the specific case of SHM, it provides end-users with information about the physical health of the structures over time and can potentially identify structural damages, which can lead to predictive maintenance and the avoidance of accidents.
- **Device Toolchain**: it supports the management of physical devices of the system. The toolchain utilizes device metadata to support system administrators with information about device location, capabilities, version, and

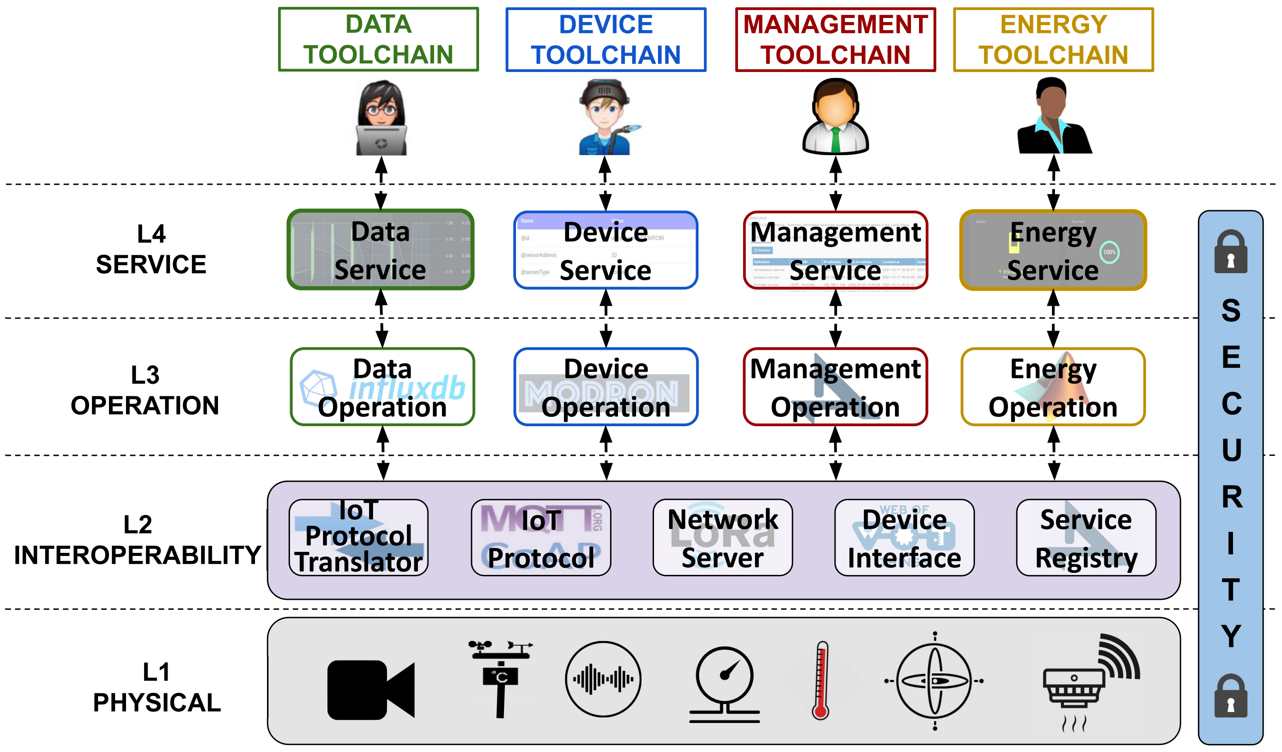


FIGURE 2. Generic toolchain architecture for condition monitoring.

current configuration. Also, it allows users or tools to alter the current devices' settings in order to enhance or optimize a certain feature of the system.

- **Management Toolchain**: it supports the monitoring of the system applications. It is responsible for: orchestrating the applications in a given infrastructure – often composed of multiple cloud and edge computational nodes; performing automated maintenance tasks, such as sanity checks; providing a flexible ecosystem that enables software components to be easily integrated, updated, deprecated, and even removed.
- **Energy Toolchain**: energy is a crucial factor for IoT-based systems. Devices are numerous and designed to last for years, thus, a delicate balance exists between battery lifetime and device operation. Improper use of the energy in the system can lead up to a meaningful increase in costs related to maintenance. We leverage the energy management of the system bringing it to the application level. The energy toolchain monitors the energy harvesters, the consumption of each device, and the current and future environmental conditions that may impact the energy, allowing rapid response, optimal assessment, and leading to more efficient use of the system energy.

The high-level design of each toolchain is meant to be a guideline to assist the implementation of IoT-based systems for the CM domain and can be accomplished with a myriad of different implementations that perform the roles described. However, in the next sections, we exemplify and illustrate how we implemented each toolchain in the scope of our use case.

IV. THE SHM PILOT: DEVICES AND INTEROPERABILITY

As an instance of the above described toolchain architecture, we present here its concrete implementation into the SHM pilot use case within the Arrowhead Tools project, reported in Figure 3.

The baseline of the use case features a structure to monitor (*e.g.*, a bridge or a building) and a set of inertial sensors for SHM permanently attached to it. Each sensor cluster is managed by a Cluster Head (CH) that acts as a sink for the data and interacts with any external actor. Sensors are abstracted into Web Things (WTs) in order to be accessible via the Web, following the W3C WoT standard [8]. Other legacy sensors, in our case a gas sensor, may be added. Compliance between the WoT and Arrowhead is achieved via a tool called WAE. All elements of our pilot that we identify with L1 and L2 of our toolchain architecture are described in this section.

We then developed a number of engineering tools, organized into our four distinct toolchains: (i) the Data Toolchain operates through MODRON, a visualization and management dashboard, (ii) the Device Toolchain operates through a number of tools that are able to dynamically and autonomously change sensor properties (*e.g.*, the sampling frequency, the duty cycle, etc.), (iii) the Management Toolchain integrates

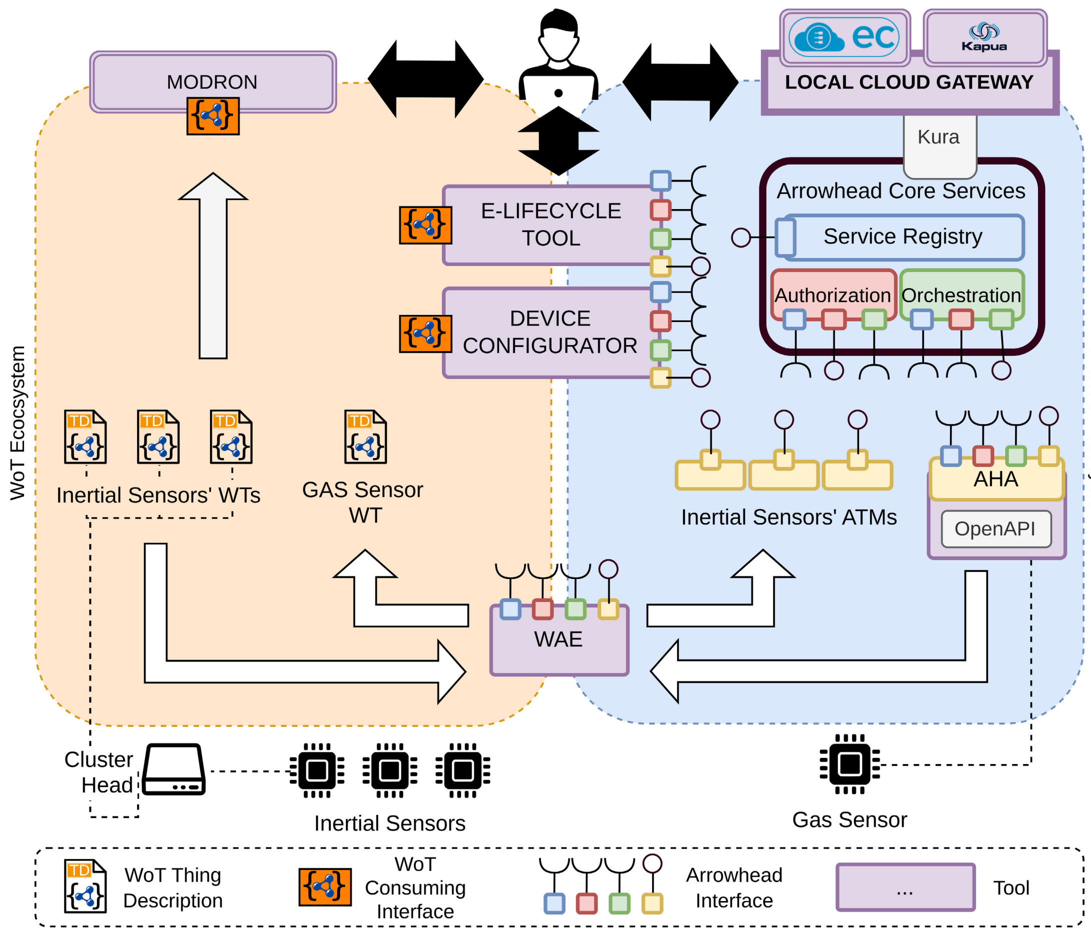


FIGURE 3. Toolchain architecture of the whole System-of-Systems, with a focus on the separation and the interoperability between the Arrowhead and the WoT ecosystems.

the scenario with the frameworks Eclipse Kura[4] and Eclipse Kapua,[5] enabling the remote management of multiple local clouds at the same time without having to interact with each of them separately, (iv) the Energy Toolchain operates through equipping each sensing device with an energy harvester and introduces tools that enable the management of energy in the application level, allowing to minimize – or, potentially, zero – the battery depletion. The description of each toolchain is presented in Section V.

All the interactions taking place among tools and application systems are mediated and overseen by an instance of the Arrowhead Core Services deployed within the local cloud. We assume that every monitored structure corresponds to a single local cloud, while multiple of them are managed through Eclipse Kapua, which is able to reach different instances of the Core Systems at the same time.

A. PHYSICAL LAYER

We consider the baseline of the environment as a physical structure to monitor and a set of monitoring sensors deployed on it. Furthermore, we can assume the structure to be in a poorly connected environment, in which providing cable connection to all the sensors is hard, therefore edge devices communicate with each other via wireless connections and are powered by batteries. Physical layer devices employed in the pilot are either connected to a sensor network tailored for SHM – described in Section IV-A1 – or third-party devices not easily integrated to an already deployed network, as the gas sensor detailed in Section IV-A2.

1) SHM SENSOR NETWORK

The adopted monitoring network consists of intelligent sensor nodes (SN) and related network interfaces designed within the Intelligent Sensor System Lab of the University of Bologna. Each peripheral device is a micro-system unit realized according to a five-building block architecture comprising [20], [21]: (i) one sensing element, *i.e.*, the MEMS-based iNemo LSM6DSL inertial measurement capable of acquiring, at the same time, triaxial accelerations and triaxial angular velocities in a broad frequency range, (ii) an STM32F303 microcontroller unit equipped with a floating point unit for optimized digital signal processing functionalities enabling

[4]https://www.eclipse.org/kura/
[5]https://www.eclipse.org/kapua/

sensor-near data analytics, (iii) a voltage regulator, which is necessary to fix the voltage reference of the node, (iv) an external RAM memory to expand the limited storage capabilities of the resource-constrained micro-processor and (v) a transceiver for RS-485-based data management and communication. Once signals are sensed by the peripheral node, information is then transmitted to the Cluster Head (CH) controller through the companion gateway network interface (referred to as SAN interface in [20] and in Figure 3).

2) THE GAS SENSOR NODE

The gas sensor node is an autonomous end-device that embeds an experimental gas sensor by STMicroelectronics [22]. The sensor is a Metal Oxide Semiconductor (MOX) sensor that provides information about volatile organic compound (VOC) type (*i.e.*, Ammonia (NH_3), Nitrous Oxide (N_2O) and Methane (CH_4) and its concentration in the air. These measurements can support SHM scenarios in a number of ways – *e.g.*, detecting the presence of heavy traffic load. In addition to the sensor, the node integrates several units : a microcontroller unit, a LoRa transceiver, and all the circuitry needed for the smart power supply system. More in detail: the MCU is an STM32L0 with ultra-low power features, the LoRa transceiver is the Semtech SX1276, and the power supply system includes a solar energy harvester, a battery management system, and a module that monitors the condition in which the energy harvester works – more on the energy harvesting aspect can be found in Section V-C in which we detail the Energy Toolchain. The low power features of the node, its power supply system, along with the long-range communication implemented by the LoRa stack makes the sensor node extremely versatile for several SHM purposes.

B. INTEROPERABILITY LAYER

Integration is one of the main factors to make advancements in digitalization and to achieve interoperability in Industry 4.0 scenarios [23]. This Section presents the interoperability tools that were designed to support the four toolchains proposed. The Arrowhead framework is a key interoperability enabler to the proposed system and its capabilities were addressed in Section II-A. Further, due to its pervasive presence in all toolchains, it was omitted for image clarity in all the following diagrams that illustrate each toolchain, namely Figure 5, Figure 6, Figure 7 and Figure 10. We detail the implemented WoT integration (Section IV-B1) and its conversion to the Arrowhead Services (Section IV-B3), also the adapter build to integrate other devices directly to the Arrowhead Framework (Section IV-B2).

1) WEB OF THINGS INTEGRATION

In order to tackle the fragmentation in the IoT landscape, the W3C consortium recently released a set of standards for the Web of Things (WoT) whose goal is to provide well-defined interfaces and guidelines for the deployment of IoT scenarios, enabling the interoperability among different systems thanks to the consolidated web technologies [9]. More in detail, each IoT system component is accurately defined through a Thing Description (TD),[6] a JSON-LD document semantically enriched that describes the affordances of the component. An affordance is intended as a fundamental property of the component that determines how the thing could possibly be used, and it can be categorized as *Property* (an internal state variable of the Thing), *Action* (a command that can be invoked on the Thing), and *Event* (a notification fired by the Thing). We followed this approach for mapping each of the devices in the SHM Sensor network to a W3C Web Thing (WT), which exposes properties like *Acceleration* and *AccelerationSamples* for reading the data and actions such as *configureFrequency* to change the configurations and behavior of the sensor [24]. The TD is semantically annotated by using the Semantic Sensor Network Ontology (SOSA).[7]

2) ARROWHEAD ADAPTER (AHA)

To take advantage of the integration that the Eclipse Arrowhead Framework offers for components, we developed a specific component called Arrowhead Adapter (AHA). The AHA has the role of overcoming components heterogeneity by making them Arrowhead compliant. It typically acts as a wrapper for the sensors, targeting both legacy and heterogeneous component integration scenarios. In this section, the functionalities of the AHA are described within the context of integrating the gas sensor detailed in Section IV-A2 within the Arrowhead framework. Figure 4 shows the reference architecture of the Gas Sensor tool, consisting of three main elements: the physical gas sensor node, the LoRa Gateway and the AHA.

The infrastructure used for the communication between the node and the server is provided by The Things Network[8] and consists of a Network Server, an Application Server, and an MQTT broker. The communication, depicted in Figure 4, is bi-directional, and the node is able to send information (uplink) and receive commands (downlink) from the server. In this sense, the role of the MQTT broker is to provide an interface between the LoRa Server and final users through a publish-subscribe mechanism.

The AHA is the element that enables compatibility between any industrial device and the Arrowhead Framework. It acts as an intermediary between the two entities by exploiting two interfaces: an HTTP/REST one enables communication with the Arrowhead Framework and acts as an endpoint for remote interaction with the node, while the other interface exploits specific protocols to interact directly with the sensor to be integrated [25]. In the particular case of the gas sensor, the functionalities to be integrated are related to the reading of the sampled values and the changing of the duty cycle in which to operate, while the protocol used

[6]https://www.w3.org/TR/wot-thing-description/
[7]https://www.w3.org/TR/vocab-ssn/
[8]https://www.thethingsnetwork.org

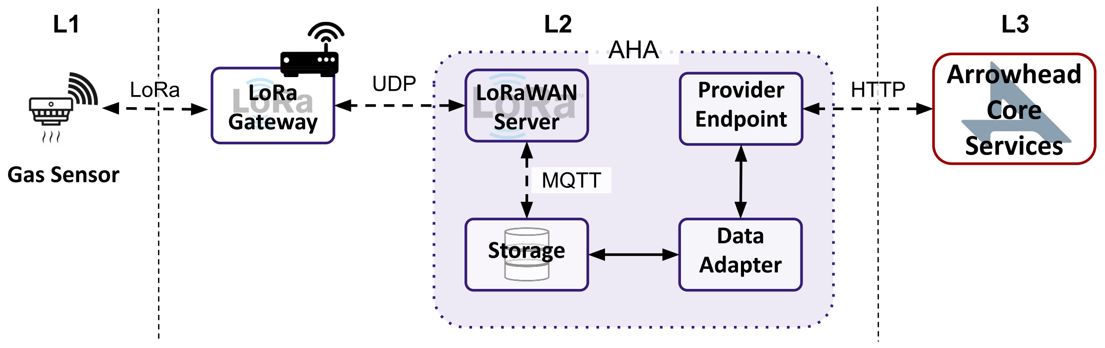


FIGURE 4. Gas Sensor Arrowhead Adapter and Network.

for communication with the node is MQTT. At startup, the adapter connects to the Arrowhead Core Services to register the two services by entering information about input parameters, then the node remains listening for requests from Arrowhead consumers. When a new data item is available, the MQTT client receives the sampling on a specific topic and saves it within the persistence layer present in the structure. The data thus stored are then made available in JSON format via the appropriate REST endpoint; here it is the adapter that handles the conversion of data from the database to the user.

3) WEB OF THINGS ARROWHEAD ENABLER (WAE)

The Web of Things Arrowhead Enabler (WAE) [26] is pivotal for the proposed toolchain, and it is an official part of the Eclipse Arrowhead framework[9] – due to Arrowhead repository naming conventions, WAE is referred as `application-skeleton-wot`. Its role is to bridge the Eclipse Arrowhead Framework with the WoT. It is challenging since the two ecosystems are heterogeneous, and there is no direct way to interconnect them. The W3C WoT defined a strict interface to interact with WTs, however, no methods or guidelines allow the conversion of dissonant interfaces to the defined standard. On the other hand, the Arrowhead Framework does not define any standard APIs and can interact with any web service that exposes a RESTful interface.

WAE performs two critical operations: (i) the discovery and integration of WTs to the Arrowhead Framework; (ii) The automatic conversion of Arrowhead services' RESTful interfaces – provided their OpenAPI Specification (OAS) – into a Thing Description and the procedures to instantiate the translated description in a WT. The first operation integrates WTs towards the Arrowhead Framework by monitoring both a Thing Directory and the Arrowhead Service Registry. If an inconsistency is detected, it means a new device was removed, edited, or added. Thus, the WAE executes the correspondent operation within the Service Registry. The second operation implies the WAE monitoring the Arrowhead Service Registry for new web services. Once one is identified, WAE translates its REST API (the translation

[9]https://github.com/arrowhead-f/application-skeleton-wot

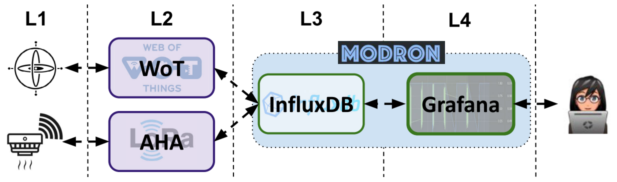


FIGURE 5. Data Toolchain.

technique is described in [27]) to a Thing Description and deploys a WT that acts as a proxy of the actual service.

V. THE SHM PILOT: TOOLCHAINS

This section presents the implementation of the four toolchains described in Section III. Each subsection details the operation of each tool that composes the toolchain.

The toolchain deployed in the Pilot is modular. Hence, they can be deployed independently from each other. However, it is common for them to be deployed together since their functionalities are complementary. Once the toolchain that enables IoT data to reach applications is in place, it is natural to develop features to easily manage devices – including their energy capabilities – and applications. Consequently, many applications will provide functionalities that are useful and intended for multiple toolchains. This is the case of MODRON, which integrates the Data Toolchain and the Device Toolchain.

A. DATA TOOLCHAIN

The Data Toolchain is the main functional toolchain in the use case, as its goal is to acquire data from the sensors deployed on the structure and report it to the internet for analysis. The toolchain includes the baseline application systems (*i.e.*, the SHM sensor network), the gas sensor, AHA, the WoT, and MODRON time-series storage and visualization tools.

MODRON: MODRON [10] is a specialized framework for managing SHM data: from saving the measurements to the database to processing and visualizing them. The system architecture consists of two macro-blocks that play a specific role within the Data Toolchain. The first block is an L3 tool that specializes in collecting and saving data acquired from

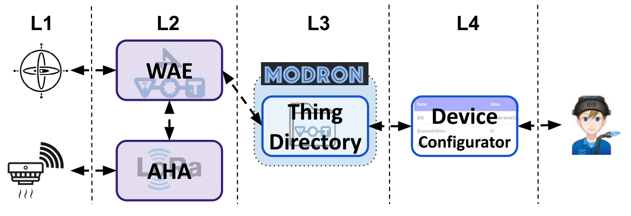


FIGURE 6. Device Toolchain.

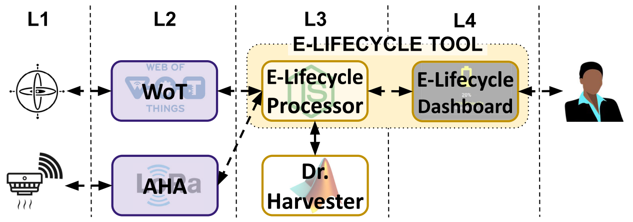


FIGURE 7. Energy Toolchain.

interoperability tools. Specifically, there are multiple micro-services responsible for querying the devices constantly. Raw data retrieved in this way is sent to a first time-series database – *i.e.*, InfluxDB[10]–, here they are kept in their original state, without applying any transformation on them. In parallel, a second pipeline processes the raw data by performing signal processing techniques for structural integrity evaluation. The second block is an L4 tool containing all the user-oriented services for interacting with the collected data. MODRON exposes two primary tools to the user: *(i)* the Data Aggregator, which allows the user to combine and perform some simple operations on the data just before displaying them; *(ii)* the Data Plotter, providing various dashboards with multiple chart options and the ability to export them in different data formats.

B. DEVICE TOOLCHAIN

The Device Toolchain enables a user of the use case to interact directly with the devices and issue them with commands that change their operating conditions at runtime. The toolchain includes the baseline application systems (*i.e.*, the SHM sensor network), the gas sensor, AHA, the WAE, the MODRON Thing Directory, and the Device Configurator.

1) MODRON THING DIRECTORY

The MODRON Thing Directory is a service dedicated to managing WoT TDs. While this component is not a fully W3C Discovery specification-compliant directory,[11] it exposes two APIs – *e.g.*, REST and GraphQL – to perform all CRUD and search operations necessary for descriptor manipulation. In detail, the Search API allows the discovery of devices through semantically enriched query languages (*e.g.*, JSONPath[12]) queries. This powerful feature gives the user the ability to perform a semantic search by taking full advantage of the metadata present on the device descriptor.

2) DEVICE CONFIGURATOR

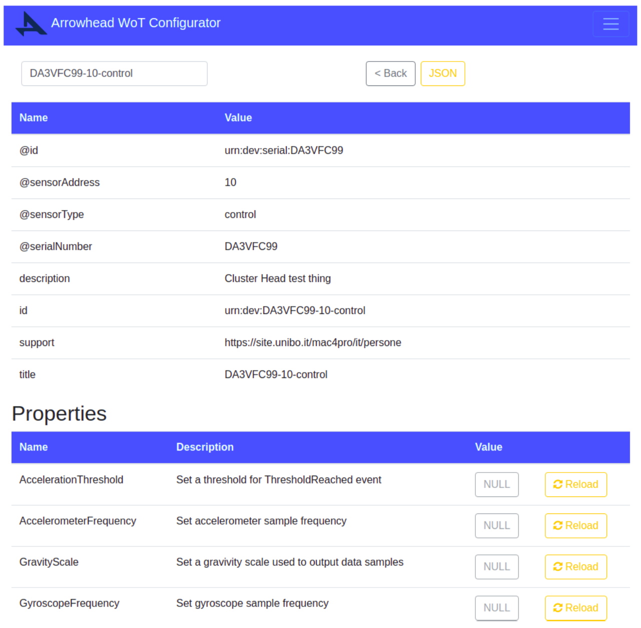
We expect real IoT-based SHM systems to have numerous and heterogeneous devices. Even with multiple tools that enable interoperability, it can be tough to keep track of all devices. Further, knowing all devices' capabilities, values, and interfaces at all times is a herculean task that makes altering a device setting or configuring a new feature very time-consuming; this scales with the complexity of the deployed system. In order to streamline this process, we designed and developed an open source tool[13] – referred to here as the Device Configurator – that provides a friendly browser-based user interface for managing WTs within the Arrowhead ecosystem, as illustrated by Figure 8. It allows users to find devices in their system, alter their properties and trigger their actions. One vital feature is its plug-an-play capability to detect new devices and automatically show them since it constantly monitors the Thing Directory where the WoT TDs are located.

C. ENERGY TOOLCHAIN

Wireless sensors are one of the key elements for many IoT applications because they are easy to install and simple to connect to existing networks and infrastructures. One of the main hardware enhancements to the initial baseline is a shift from cable-connected edge devices to wireless ones. However, the latter are traditionally battery-powered, and this is the main limiting factor to their extensive use in real applications [28]. To be compliant with application-specific requirements (*e.g.*, data sampling and transmission rate), the node power consumption often does not guarantee an acceptable recharge/replacement rate of the battery leading to not sustainable maintenance costs and too frequent short interruption of service. The adoption of Energy Harvesting (EH) solutions can help to extend the battery life or even obtain battery-less autonomous devices, because the edge devices (*e.g.*, Gas Sensor and SHM sensor network) become able to gather energy directly from the environment where they operate, by transforming energy from light, thermal gradients or mechanical vibrations in electrical energy. Unfortunately, the design of an energy harvesting module is usually a complex and strictly application-specific task requiring a lot of design efforts in terms of both time and costs. It is in this context that an EH toolchain has been developed, composed of the energy harvesters and a number of software tools: Dr. Harvester and the E-Lifecycle Tool.

An EH circuit is comprised of three main functional blocks: i) the energy source module (*e.g.*, solar panel) to collect energy from the environment where the device works; ii) the energy conversion and management circuit to efficiently power supply the device by using the collected energy;

[10]https://www.influxdata.com/
[11]https://w3c.github.io/wot-discovery
[12]https://www.ietf.org/archive/id/draft-goessner-dispatch-jsonpath-00.html
[13]https://github.com/UniBO-PRISMLab/wot-configurator



DA3VFC99-10-control

Name	Value
@id	urn:dev:serial:DA3VFC99
@sensorAddress	10
@sensorType	control
@serialNumber	DA3VFC99
description	Cluster Head test thing
id	urn:dev:DA3VFC99-10-control
support	https://site.unibo.it/mac4pro/it/persone
title	DA3VFC99-10-control

Properties

Name	Description	Value	
AccelerationThreshold	Set a threshold for ThresholdReached event	NULL	Reload
AccelerometerFrequency	Set accelerometer sample frequency	NULL	Reload
GravityScale	Set a gravivity scale used to output data samples	NULL	Reload
GyroscopeFrequency	Set gyroscope sample frequency	NULL	Reload

FIGURE 8. **A screenshot of the Configurator tool.**

iii) the energy buffer (*e.g.*, rechargeable battery or supercap) to store the collected energy. As mentioned before, customization based on application-specific requirements is needed to obtain effective EH solutions. For example, the two considered case studies – the Gas Sensor and the SHM sensor network (entirely powered by the CH, which is hosting the harvester) – have very different power consumption and consequently, they need two different EH circuits. In this work, the design effort to obtain suitable solutions for both case studies has been reduced thanks to the developed Dr. Harvester tool.

1) DR. HARVESTER

The tool is comprised of a MATLAB-based core application able to interact with an electrical circuit simulator and it has been designed to be used both at design time and at operation time. It is based on a set of pre-designed EH circuits, each circuit is designed to match a given set of environmental working conditions (*i.e.*, solar irradiance, thermal gradients, and mechanical vibrations) and a range of load power consumption (*i.e.*, power consumption of the edge device). In this case, LTSpice[14] from Analog Devices has been used as an electrical circuit simulator, but it is worth noting that it is very easy to adapt the tool to any other PSpice engine-based simulator because the main parameter passed from the MATLAB core to the electrical simulator is the EH circuit netlist and not the schematic. Basically, the design-time and operation-time versions of the tool exploit the same core, while differing in i) the main purpose; ii) the elements of the tool chain in which they interact; iii) the interaction methods.

[14]https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

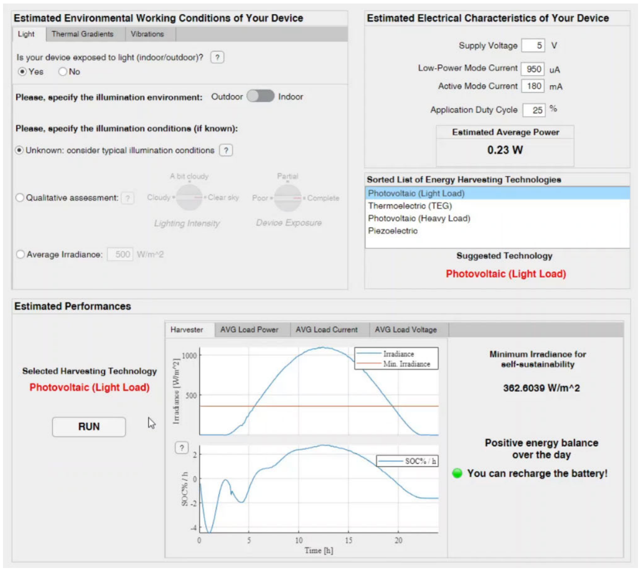
Design-Time Usage – At design time, the main goal of the tool is to help the designer to add energy harvesting capabilities to a generic electronic device (an existing one or a new one designed from scratch) with limited design efforts. At design time the tool is running locally on a PC and needs human interaction. In this case, the operating cycle of the tool is comprised of 4 steps:

1) By exploiting an intuitive Graphical User Interface (GUI), the designer specifies i) the estimated environmental working conditions of where the device that is being developed will operate; ii) the estimated electrical characteristics of the device (*i.e.*, supply voltage, current consumption, application duty cycle).
2) During the data entry, the tool runs a decision logic algorithm and provides a sorted list of the available EH solutions supported by the tool, suggesting the EH solution that best match the combination of provided environmental and electrical estimated working conditions.
3) The user selects the desired EH solution and the tool automatically runs the related circuit simulation.
4) At the end of the simulation, the tool shows in the GUI both qualitative and quantitative results about the feasibility and estimated performances of the chosen EH solution. Once the first iteration is completed, the designer can iterate the procedure by changing some input parameters (environmental and/or electrical) and evaluate a variant of the initial solution. For example, the designer can define the maximum application duty cycle allowing obtaining a fully autonomous device for a given power consumption and a range of real environmental working conditions.

An example of how the developed GUI appears at the end of a simulation is shown in Figure 9. The results refer to one of the iterations carried out during the design of the solar energy harvester embedded into the Gas Sensor.

Run-time Usage – At operation time, instead, Dr. Harvester is used to forecast the behavior of a remote-controlled edge device (*e.g.*, Gas Sensor or SHM sensor network) under the provided working conditions (real or estimated). Here the main outputs of Dr. Harvester are the remaining battery lifetime or the remaining time to fully recharge the battery depending on the simulated working conditions. All the tasks are triggered by the E-Lifecycle Tool through an Arrowhead-enabled REST interface. To decouple the interaction with other tools, Dr. Harvester was converted into an Arrowhead Service Provider. The tool, by registering services on the Arrowhead Framework of the local cloud, allows remote submissions of simulations via a REST interface and retrieves results on the remaining battery life that will then be used for optimization. Each request causes the input to be validated, which is then forwarded to the MatLab/LTSPICE process to start the job. Since a simulation might take time, it is handled asynchronously.

First, the E-Lifecycle Tool transforms sensor data in a data structure that complies with Dr. Harvester's simula-



Estimated Environmental Working Conditions of Your Device

Light | Thermal Gradients | Vibrations

Is your device exposed to light (indoor/outdoor)? ?
[✓ Yes] No

Please, specify the illumination environment: Outdoor Indoor

Please, specify the illumination conditions (if known):
[✓ Unknown: consider typical illumination conditions] ?
Qualitative assessment: ? — Lighting Intensity: Cloudy / A bit cloudy / Clear sky — Device Exposure: Poor / Partial / Complete
Average Irradiance: 500 W/m^2

Estimated Electrical Characteristics of Your Device

Supply Voltage: 5 V
Low-Power Mode Current: 950 uA
Active Mode Current: 180 mA
Application Duty Cycle: 25 %

Estimated Average Power: 0.23 W

Sorted List of Energy Harvesting Technologies
- Photovoltaic (Light Load)
- Thermoelectric (TEG)
- Photovoltaic (Heavy Load)
- Piezoelectric

Suggested Technology: Photovoltaic (Light Load)

Estimated Performances

Selected Harvesting Technology: Photovoltaic (Light Load)

RUN

Harvester | AVG Load Power | AVG Load Current | AVG Load Voltage

Minimum Irradiance for self-sustainability: 362.6039 W/m^2

Positive energy balance over the day
● You can recharge the battery!

FIGURE 9. GUI of the design time version of Dr. Harvester tool. In this case, under the provided environmental and electrical conditions, the simulated EH circuit is able to fully sustain the Gas Sensor with a minimum irradiance of about 360 W/m^2.

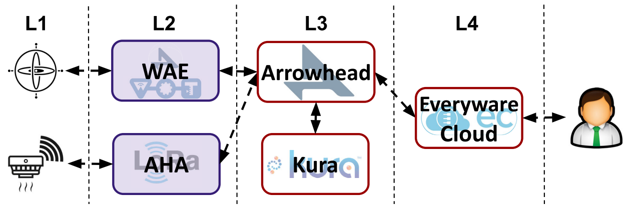


FIGURE 10. Management Toolchain.

tion input, gathering the EH system meta-information (*e.g.*, battery storage capacity) and the working conditions of interest retrieved from the edge device (*i.e.*, the actual electrical and environmental working conditions and the actual device's battery state of charge). Second, the E-Lifecycle Tool launches Dr. Harvester which executes the simulation of the EH circuit corresponding to the input parameters. Third, Dr. Harvester generates an output containing the simulation results. More precisely, the output generated by Dr. Harvester contains the battery status (*i.e.*, the battery is actually charging/discharging), the rate of variation of the battery state-of-charge and the simulation status to inform the E-Lifecycle Tool that the simulation has been completed correctly or not, and consequently if the results are valid or have to be discarded. It is worth noting that the software architecture of Dr. Harvester follows a modular approach with an independent branch for each EH circuit, and stand-alone models for both energy transducers (*i.e.*, solar panel, thermoelectric generators, cantilever-based piezoelectric devices) and batteries (*i.e.*, Li-Ion, Li-Poly, etc.). In this way, it is easy to add new supported EH circuits to the tool widening the range of covered IoT or IIoT applications.

2) E-LIFECYCLE TOOL

The Dr. Harvester tool allows executing simulations during the system operation time about the current environmental conditions and device state. However, the raw result of a single simulation is not useful for the system administrator. To leverage that, we implemented a new tool called the E-Lifecycle Tool. The tool comprises a Web interface, the E-Lifecycle Dashboard, that showcases in an intuitive form each device's current status – highlighting its battery percentage and duration – and the current solar irradiance. Such data is retrieved by the E-Lifecycle Processor, a backend module of the E-Lifecycle Tool. The E-Lifecycle Processor gathers battery and duty cycle data from the sensors and translates them into the right format for Dr. Harvester and vice-versa. Further, we allow users to experiment through the dashboard with different duty cycles for a given device and obtain the prediction of the battery duration for that configuration. Via the dashboard, the operator can then alter the device's duty cycle, by issuing the command to the E-Lifecycle Processor. Like the Device Configurator presented in Section V-B2, the E-Lifecycle Tool discovers new things automatically – powered by the Interoperability layer tools – and is capable of interacting with heterogeneous devices – from the legacy gas sensor to the SHM sensor network. As each simulation can take time to execute, we deployed a caching layer within the E-Lifecycle Processor. This way, we decrease the significant waiting time of future requests – enhancing user experience – and avoid redundant computation on the server side [29]. The E-Lifecycle Tool is available as an open–source application.[15]

D. MANAGEMENT TOOLCHAIN

The Management Toolchain enables a maintenance operator to have access to the whole fleet of SHM deployments, in order to perform sanity checks and other maintenance operations. The toolchain includes the baseline application systems (*i.e.*, the SHM sensor network), the gas sensor, AHA, the WAE, the Local Cloud Gateway, the Arrowhead Framework, Kura, Kapua, and the Fleet Management System (we address a multitude of SHM deployments as a "fleet").

1) LOCAL CLOUD GATEWAY

The Local Cloud Gateway plays a central role on the edge because it represents the HW/SW stack required to implement and run a secure, service-oriented, and remotely manageable local cloud. The hardware platform is a Eurotech Reliagate 20-25 (see Figure 11), a high-performance, globally certified, multi-service IoT edge gateway for industrial and rugged applications. It features up to four cores, soldered-down ECC RAM and storage, extended operating temperature support, wide range power supply, isolated and protected I/O interfaces, and customizable connectivity

[15]https://github.com/UniBO-PRISMLab/arrowhead-optimizer

FIGURE 11. The Local Cloud Gateway HW platform.

options, providing a very rich and heterogeneous set of interfaces and functionalities to satisfy the typical requirements of SoS remote monitoring and control applications. On this hardware platform, the local cloud software stack runs, including the Eclipse Arrowhead framework and the Eclipse Kura IoT framework, which have been integrated and extended to allow the remote monitoring and control of a local cloud. Eclipse Kura is an open-source IoT framework based on OSGi[16] and intended to provide full control on the data acquisition from the field, on the gateway hardware and software, on the cloud connectivity and intended to offer a rich set of service-oriented APIs to hide the low-level hardware and software details, simplifying the application development. Finally, in addition to enabling system integration on the edge, the Local Cloud Gateway provides the functionalities required to enable the integration at SoS level of a fleet of Local Clouds deployed on the edge (as outlined in detail in the "Fleet Management Tool" description). SoS integration is managed through Eclipse Kapua, the open-source version of Eurotech Everyware Cloud. Kapua is a modular IoT cloud platform intended to manage and integrate fleets of devices, their data, and IoT services for IoT applications.

2) DEPLOYMENT OF THE ARROWHEAD CORE SERVICES

The toolchain architecture envisions the gateway as the Local Cloud manager capable of handling access to all devices within it. Through light-weighted container virtualization, it was possible to install the Arrowhead core services locally dealing with the deployment of the Service Registry, Authorisation, and Orchestration components. The WAE made it possible to register all the SHM sensors, natively born under the WoT paradigm, within the Service Registry while, on the other hand, the AHA took care of the registration of the gas sensor. Furthermore, the role of the gateway consists in the implementation of all the edge component that allows data to be read from the physical sensors and sent to the cloud for data analytics or visualization operations. To do this, the Kura framework enabled the creation of bundles that were, then, specialized according to the different sensors to be interfaced. Each of them is configured based on a properties file, containing both information about where the core services are maintained and also data about the specific sensor to be queried (*e.g.*, the definition of the service exposed by the sensor) and the polling period. The first step in the bundle is to verify that the services are active and to perform automatic discovery of the requested service to check that it exists; once satisfied, one can proceed to request the sensor access data from the Orchestration core system. Once the data is present, a periodic task is scheduled with the purpose of querying the sensor, based on the polling parameter, and retrieving its significant fields. The Kura administration page provides access to the configuration of each bundle by allowing both the polling period and the sensor to be interfaced to be varied. The last step is to send the data to the cloud, respectively, the Everyware Cloud and the MODRON framework, for final visualization.

[16]https://www.osgi.org/

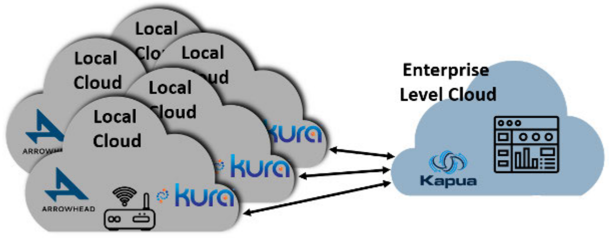


FIGURE 12. The local clouds fleet management.

3) LOCAL CLOUDS FLEET MANAGEMENT TOOL

This cloud-based tool operates at SoS level and is responsible for the integration and remote monitoring of a fleet of Local Clouds. The monitoring of a fleet of Local Clouds implies the capability to communicate with each of the Local Clouds on the field and to be able to send their information to an enterprise-level cloud platform where it can be displayed and monitored. Each of the Local Clouds has the capability to connect to the enterprise-level cloud through Eclipse Kura. On the enterprise-level cloud, Eclipse Kapua collects all the information from the fleet of Local Clouds and makes it available through a web console (see Figure 12).

The integration between Kura and the Arrowhead Framework is based on the development of a new core system (as depicted in Figure 13), enabling the remote collection and monitoring of the Local Cloud information [30]. The new system has been called Remote Management and it exposes a single REST endpoint which is the public echo endpoint, having the sole purpose of checking if the core system is up and running. The Remote Management communicates directly with the Service Registry and the Orchestrator systems through their management endpoints and retrieves information about the registered consumer and provider systems, together with the services they made available. Remote Management acquires also the Orchestration Store rules which establish the appropriate providers for the consumers. The role of Remote Management is then to populate the gateway's

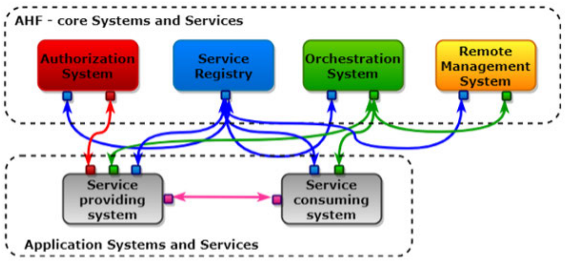
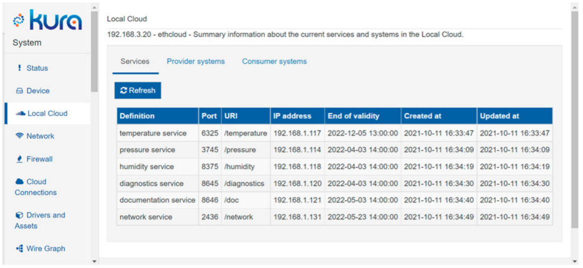


FIGURE 13. The new Arrowhead framework core system.



Local Cloud

192.168.3.20 - ethcloud - Summary information about the current services and systems in the Local Cloud.

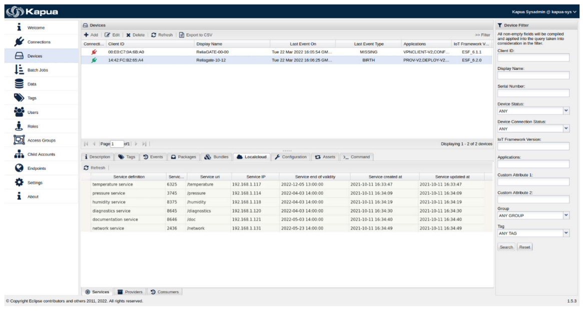
Definition	Port	URI	IP address	End of validity	Created at	Updated at
temperature service	6325	/temperature	192.168.1.117	2022-12-05 13:00:00	2021-10-11 16:33:47	2021-10-11 16:33:47
pressure service	3745	/pressure	192.168.1.114	2022-04-03 14:00:00	2021-10-11 16:34:09	2021-10-11 16:34:09
humidity service	8375	/humidity	192.168.1.118	2022-04-03 14:00:00	2021-10-11 16:34:19	2021-10-11 16:34:19
diagnostics service	8645	/diagnostics	192.168.1.120	2022-04-03 14:00:00	2021-10-11 16:34:30	2021-10-11 16:34:30
documentation service	8646	/doc	192.168.1.121	2022-05-03 14:00:00	2021-10-11 16:34:40	2021-10-11 16:34:40
network service	2436	/network	192.168.1.131	2022-05-23 14:00:00	2021-10-11 16:34:49	2021-10-11 16:34:49

FIGURE 14. Eclipse Kura local web page for local cloud monitoring.

internal database with the Local Cloud information acquired from the Service Registry and from the Orchestration system. The advantage of this solution is that the new system (Remote Management) has been integrated at the Arrowhead Core System level, enabling it to seamlessly access endpoint management information from the Service Registry and from the Orchestration system. Additionally, this approach has the advantage that the monitoring of the Local Cloud is fully integrated within Kura, avoiding any need for external tools or dashboards for its visualization; besides the standard device monitoring offered by Kura, the extension allows to access the full list of providers, consumers, and services. Indeed, Eclipse Kura accesses directly the gateway's database to retrieve the data and has been extended to visualize it on the Local Cloud Gateway web interface, while delivering it also to the enterprise-level cloud (see Figure 14). Eclipse Kapua has been extended to collect the information sent by each instance of the Local Cloud composing the fleet. Figure 15 shows the new section on Kapua's web console, under the "Devices" menu, devoted to Local Cloud monitoring. The new section provides the list of devices connected to the cloud (green plug symbol). Each of these devices corresponds to a physical Local Cloud Gateway, which is a Kura instance that is connected to a Local Cloud through the Remote Management system.

VI. RESULTS AND DISCUSSION

The effectiveness of the SHM application is validated on an existing use case, which consists of an experimental model of a bridge (Figure 16) located at the Laboratory of Structural and Geotechnical Engineering (LISG) of the University of Bologna. The structure represents a 1:4 scale reproduction of a real composite steel-concrete bridge crossing the A14 highway in Italy, near the city of Bologna. The scale replica preserves the materials and their properties; changes concern only the dimensions. More information about the model bridge manufacturing process and material properties can be found in [31].



Connecti...	Client ID	Display Name	Last Event On	Last Event Type	Applications	IoT Framework V...
	00:E0:C7:0A:6B:A0	ReliaGATE-00-00	Tue 22 Mar 2022 16:05:54 GM...	MISSING	VPNCLIENT-V2,CONF...	ESF_6.1.1
	14:42:FC:B2:65:A4	Reliagate-10-12	Tue 22 Mar 2022 16:06:25 GM...	BIRTH	PROV-V2,DEPLOY-V2...	ESF_6.2.0

Page 1 of1

Displaying 1 - 2 of 2 devices

Service definition	Servic...	Service uri	Service IP	Service end of validity	Service created at	Service updated at
temperature service	6325	/temperature	192.168.1.117	2022-12-05 13:00:00	2021-10-11 16:33:47	2021-10-11 16:33:47
pressure service	3745	/pressure	192.168.1.114	2022-04-03 14:00:00	2021-10-11 16:34:09	2021-10-11 16:34:09
humidity service	8375	/humidity	192.168.1.118	2022-04-03 14:00:00	2021-10-11 16:34:19	2021-10-11 16:34:19
diagnostics service	8645	/diagnostics	192.168.1.120	2022-04-03 14:00:00	2021-10-11 16:34:30	2021-10-11 16:34:30
documentation service	8646	/doc	192.168.1.121	2022-05-03 14:00:00	2021-10-11 16:34:40	2021-10-11 16:34:40
network service	2436	/network	192.168.1.131	2022-05-23 14:00:00	2021-10-11 16:34:49	2021-10-11 16:34:49

Device Filter

All non-empty fields will be compiled and applied into the query taken into consideration in the filter.

Client ID:	
Display Name:	
Serial Number:	
Device Status:	ANY
Device Connection Status:	ANY
IoT Framework Version:	
Applications:	
Custom Attribute 1:	
Custom Attribute 2:	
Group	ANY GROUP
Tag	ANY TAG

© Copyright Eclipse contributors and others 2011, 2022. All rights reserved.

1.5.3

FIGURE 15. Eclipse Kapua remote monitoring of local clouds.

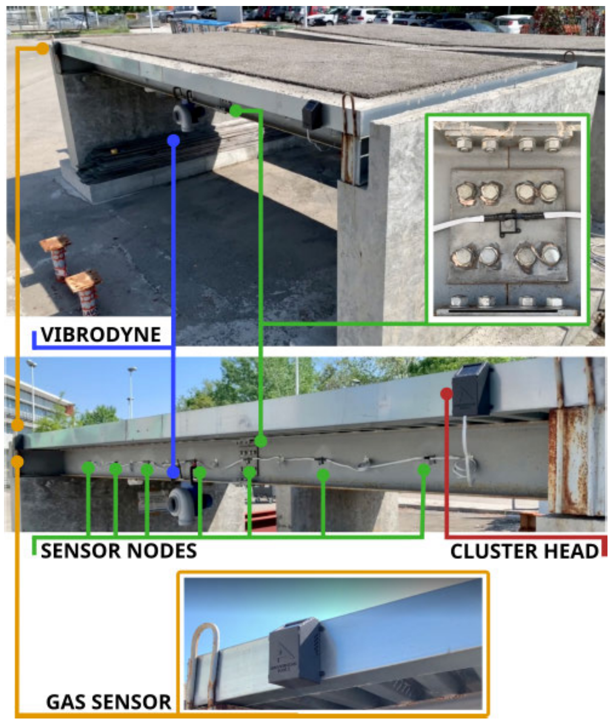
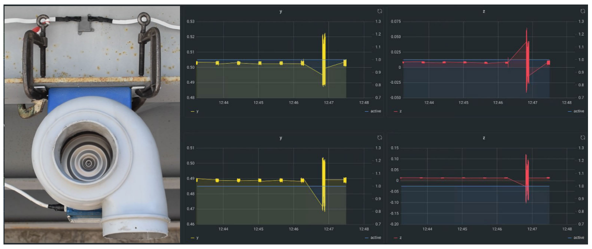
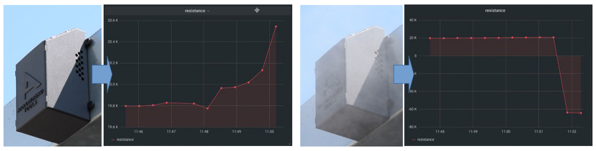


FIGURE 16. Bridge model under test.

The deployed Structural Health Monitoring sensor network and its main components are displayed in the middle callout of Figure 16, in which one CH (red point) and seven SNs (green points) are noticeable, placed at strategic positions for vibration analysis, *i.e.*, in correspondence of the anti-nodal points of the first modal components. A snapshot of one SN installed at the bridge deck joint has also been enclosed in the upper part of Figure 16. Besides, a vibrodyne (Figure 16 blue point) has been applied to the bridge in order to apply a variable-intensity harmonic excitation up to its power supply settings. In addition to the SHM network outlined above, the deployment includes the gas sensor (Figure 16 orange points). Both the gas sensor and the CH are powered by an EH circuit attached to a solar panel. The purpose of this section is to illustrate how we highlighted the accomplishment of the three

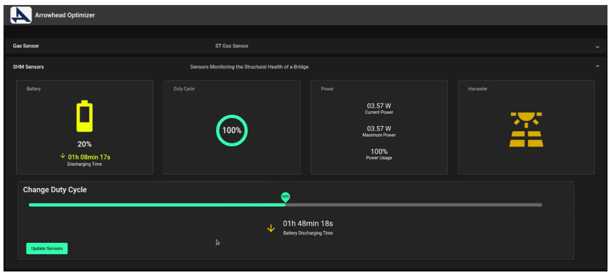


(a) Sensor readings of the SHM sensors on MODRON with the vibrodyne on.

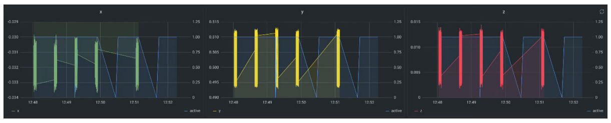


(b) Sensor readings of the gas sensor on MODRON.

FIGURE 17. The upper figure shows the accelerometer bursts that change after the vibrodyne is turned on. The lower figure shows how the data from the gas sensor changes before and after the gas sensor is sprayed with gas.



(a) The E-Lifecycle Dashboard.



(b) Sensor readings and duty cycles on MODRON.

FIGURE 18. Screenshots showing the E-Lifecycle Dashboard (above) during the operation of changing the duty cycle of the SHM sensors from 100% to 50%. The result of the duty cycle change is shown through MODRON on three axes of a single SHM sensor (below): the blue line identifies the wakeup intervals of the sensors. When the line is not set to 1 the sensors do not perform any read operation.

project objectives set within the Arrowhead Tools project through the combined usage of the four toolchains.

A. INTEGRATION WITH LEGACY SYSTEMS

The four toolchains are composed of several tools that altogether concur in achieving the first project objective: the seamless interoperability and the integration of legacy systems. This is most evident for the tools that compose the Data and the Device Toolchains. The demo environment illustrated above contains a number of off-the-shelf legacy devices and systems that need an integration step in order to be fully operational in our heterogeneous scenario. The SHM sensors were originally abstracted into WTs, however, they lacked interoperability towards the Arrowhead Local clouds. On the other hand, the gas sensor was integrated within Arrowhead via a dedicated adapter. However, only with the WAE and the integration with Kura, we achieved a fully interoperable layer, so that the interaction with the devices from the upper layers is totally agnostic of the standard they comply with. The validation of this aspect has taken place by means of the sensor values shown through the Grafana dashboard of MODRON via the Data Toolchain. Figure 17 shows the sensor values reported into said dashboard. In particular, the upper figure shows how we solicit sensor readings by turning on the vibrodyne and, thus, injecting vibrations into the structure. These are captured by the accelerometers and reported in one of the last bursts on their Y and Z axes, also visible in the picture (the X axis is not very significant as the bridge does not vibrate much longitudinally). The exact same system is used to detect variations in the concentration of gas, as captured by the gas sensor in the lower part of the figure. In particular, we show how injecting gas in the environment (in this case through a spray can) causes the sensor resistance to dramatically drop – note that the readings in absence of gas appear to variate, however, this is negligible as the scales on the two Y-axes are very different. The validation shows a small real-world fully operational system, however, the main advantage here is the interoperability: in fact, adding more physical sensors or other systems does not cause the complexity to increase, as the ''hub'' systems such as MODRON do not need any additional interface to cope with onboarding processes, rather they take place seamlessly.

B. REDUCTION OF ENGINEERING COSTS

All illustrated toolchains increase the automation of the baseline as well as reduce the need for human intervention, which implies a significant reduction in the cost of manual effort. However, the major evidence of cost reduction was brought in by the Energy Toolchain, presented in Section V-C, which introduces a number of decision-making tools that support the engineering process both at design time and runtime, greatly simplifying the interaction with the physical components as well as guiding their parameters towards an optimal resource consumption. Therefore, this toolchain is the main enabler for the reduction of engineering costs. First of all, controlling the data rate at which information is gathered is of utmost importance in energy-efficient monitoring scenarios, especially when this parameter can be changed remotely. In [24] we have showcased the capabilities of the Device Configurator in such a sense. In particular, each of the sensor nodes has been equipped with a WoT writable property through which such a parameter can be changed. Then, the Device Configurator has been set in order to be able to operate on such a property by exposing a number of suitable values to the system manager. The approach has been validated and demonstrated on a test frame building where the sampling frequency was decreased from 813 Hz to 416 Hz. We validated the combined impact of the E-Lifecycle Tool and Dr. Harvester on our test environment. Figure 18 shows the E-Lifecycle Dashboard with two tabs: one for the Gas Sensor and one for the SHM sensors. Both behave in the exact same way, thanks to the tools at the Interoperability Layer, that homogenized the access to different systems. In this case, the operator is changing the duty cycle of the SHM sensors from 100% to 50% obtaining a gain in the battery lifetime prospect, as calculated by Dr. Harvester. Once the operator considers all the options and changes the duty cycle, then the E-Lifecycle Processor performs such a change over the sensors via their WoT writable property, much like the Device Configurator does for the sampling frequency. The bottom of Figure 18 shows the Grafana-based dashboard of MODRON which depicts the sensor bursts and their wakeup/sleep state via the blue line. It is evident how bursts that would be occurring when the line is not set to 1 are instead not performed.

This demonstration shows the applicability of the described toolchain to a real system in a small use case, but also it shows how the combined action of the E-Lifecycle Tool, the Device Configurator, and Dr. Harvester allows the implementation of a decision support system operating at system level that can be applied in many IoT and IIoT emerging applications and contributes significantly to the engineering, deploying, commissioning and maintenance costs reduction. The possibility to forecast the amount of energy available at a given time and for a given set of working conditions (environmental and electrical) and reconfigure all the devices of a remote complex system is the key factor to obtaining a context-aware dynamic optimization of the performance of both single edge device and the whole system. Moreover, a fundamental feature that guided the design of all applications that composed the toolchain is that when the system scales up – *i.e.*, new devices are added –, the complexity of the operations that the tools enable should not equally increase. A good illustration of such behavior is perceived in the two GUI-based tools developed: the E-Lifecycle Dashboard and the Device Configurator. The two tools which implement a plug-and-play feel are designed to aid system administrators in managing devices. The operations enabled by them can be manually executed, but specific knowledge of the system and additional time are required – both factors translate to added costs. Although managing a reduced number of devices is feasible, that is not true for more elaborate systems. Hence, the added complexity of changes in IoT-based systems does not impact the tool's operation, mitigating the system's overall complexity and cost.

C. SYSTEM OF SYSTEMS INTEGRATION

SoS integration represents a challenging task, due to the complexity of the use case, the heterogeneity of the hardware devices involved, the adopted connectivity protocols, the different applications running on the devices, etc. The approach to SoS integration adopted in the proposed solution relies on the concept of SOA and on the integration with open IoT platforms (Eclipse Kura and Kapua), supported by a large community of developers and widely adopted in industrial applications. The Eclipse Arrowhead framework represents an efficient solution to manage the interaction of heterogeneous devices on the edge with an SOA architecture and introduce an interoperable way to securely share data and functionalities in the form of services through the Local Cloud. At this level, the system integration is focused on the edge and covers the devices installed on the bridge. The integration between the Eclipse Arrowhead framework and Eclipse Kura increases the functionalities available on the edge, providing extended support for field protocols, full control of the Local Cloud gateway, advanced tools and libraries to develop the application on the edge, and a rich set of connectors to the most popular cloud platforms. The connectivity to the cloud, specifically with Eclipse Kapua, and the remote management and control functionalities offered by Kura enable the SoS-level integration: the remote systems are represented by the bridges with their remote Local Clouds, while the SoS is represented by the "fleet" of structures/local clouds. The extension and integration with Eclipse Kapua is the key element for the existence of the SoS, allowing it to monitor and control it from an enterprise-level cloud and a specific web console. From the console, a single operator can monitor and control every single bridge, including the data streams, the status of the system/services installed on the bridge, and the application running on the Local Cloud Gateway, with the possibility of remotely resetting, restarting, and updating it. Finally, Eclipse Kapua allows the interaction of the SoS with enterprise-level software (*e.g.*, software for analytics, DBMS, DMS, administration, operation, and maintenance, etc.), ensuring the inclusion of the SoS in the entire monitoring, control and maintenance value chain.

D. DISCUSSION AND TRADE-OFFS

Within this Section, we outlined how we technically met the three main project objectives, taking an important step forward in the field of CM. These milestones have rarely been reached all at once in literature, as we will see more in the subsequent Section. Our proposed architecture, while composed of several moving pieces, specifically promotes the separation of concerns: as we can appreciate from Figure 2, we separated the components into conceptual abstraction layers, as it is done commonly for complex IoT ecosystems in literature. Moreover, we also proposed a "transversal" separation – corresponding here to toolchains –, which organizes the features of the architecture into distinct pipelines. These can run independently without necessarily affecting each other, thus facilitating the job of specialized personnel. This further separation of concerns is completely novel and allows actors to span over multiple abstraction layers without the full knowledge of the system. However, this toolchain-based organization brings along a number of trade-offs that cannot be disregarded, especially in presence of multiple IoT standards, as in our use case. Specifically, the usage of a component along multiple toolchains, at times, implies the related software entity to be replicated in order to respond to different uses. It is the case of the sensor nodes for our use case, where they get abstracted into WTs, ATMs, and Kura bundles. Each of these abstractions is devoted to differentiating tasks, thus exposing only a specialized subset of functionalities that do not allow its consumers to perform unauthorized or dangerous operations. While this is a clear advantage from the safety and management point of view, we need to ensure that the mirroring entity, in our case the WAE, does not allow any misalignment of the representations of the same component across different domains. Secondly, this practice makes the number of software entities grow significantly, thus bringing scalability issues to the forefront. We believe that more work can be done in the context of Toolchains-of-Toolchains (ToT), as pointed out in [14], in terms of simplifying the workload on software entities and avoiding resource redundancy where possible.

VII. RELATED WORKS

CM assesses the integrity condition of structures and machinery, which is regarded as one of the main priorities for ensuring a safer environment. Among the very manifold application domains, this necessity is particularly claimed in the case of civil infrastructures and industrial installations, where natural and manufactured aging factors are crucial. Nonetheless, the evolving complexity inherent in the industrial and construction design process requires the monitoring system to be resilient, responsive, and robust against flaws. Satisfying all these features simultaneously poses extreme challenges, specifically within the Information and Electronics community, which is constantly triggered to provide more efficient and versatile solutions capable of integrating and orchestrating different technologies and abstraction layers. In this paper, we mainly focused on the problems arising from the high interdisciplinarity of CM. As we have seen, modern approaches require, on the one hand, a set of abstraction layers that let diverse technologies work together, on the other hand, the tools for different human actors to accomplish their dedicated tasks without affecting other pipelines.

Due to the wide variety of the actors involved, from the embedded sensors to remote cloud platforms, existing works usually focus horizontally on one or two sub-components (*e.g.*, sensing layer *vs.* data analytics), while providing only a limited understanding about the remaining elements. This is the case, for example, of the seminal work by Tokognon et al. [32], which includes an extensive overview of fundamental software-oriented problems, such as the different wireless communication and routing protocols and the data storage methodologies. Conversely, the structural characterization problem is thoroughly analyzed in [33], from the feature extraction phase up to the diagnostics and prognostics functionalities, however, all the remaining CM network components are entirely disregarded. In line with this, Lynch and Farrar [34] concentrate on the data analytics and structural inference actions as pivotal elements of the CM process. However, the IoT infrastructure and how the latter is interfaced with the other building blocks are not commented on. A more comprehensive attempt is performed in [35], where the authors examine the common layers of a typical sensor-to-cloud system. Even if the main ingredients of the CM ecosystem are described, no information about how they can be arranged is also entailed since they are presented as standalone blocks. Another work, which shows the same lack of IoT and edge processing perspective, is the one in [36] in which the authors focus on the integrity evaluation of long-span bridges.

In this manuscript, we proposed a four-layer architecture IoT architecture for CM. It is composed of four toolchains, each enabling a different feature. There is a vast literature of layer architectures for IoT-based systems; however, there is little convergence – or standards –, which led to a myriad of heterogeneous approaches and taxonomies [37]. Hence, we identify several layered IoT architectures that can be applied to CM that stand out, and we compare these to our proposed solution.

The three-layer architecture was proposed in the early development stages of IoT [38], [39], and it represents the most simple and generic definition of an IoT-based architecture. Thus, it can be implemented in all IoT application domains. It is composed of: (*i*) Perception or Sensing Layer, which includes all the sensors and acquisition devices which are necessary to collect data; (*ii*) Network Layer, which serves as the intermediate layer between the edge devices and the central unit. Among its primary functions, it is responsible for transferring data while ensuring a secure connection in a sensor–to–cloud direction; (*iii*) Application Layer: it includes all the essential services the monitoring process requires, such as data processing and visualization.

Despite its ease of implementation, such an IoT approach lacks consistency in that it cannot capture the inherent complexities of the current condition monitoring requirements. A variation of the traditional three-layer architecture for SHM was proposed by Zonzini et al. [40]. The architecture encompasses the particular features of SHM-based sensors (*e.g.*, accelerometers) and introduced computing capabilities in the network edge, which is responsible for exposing the IoT devices through an interoperable interface. The same system was further enhanced in [10], where the importance of the edge was highlighted, making it a layer of its own.

A four-layer variant of the IoT architecture has been proposed to distribute better the tasks between the architectural resources of the CM system, thus favoring the development of a more versatile and timely responsive framework. Compared to the three-layer architectures, the four-layer variant introduces a purposely–devoted *Processing* or *Support layer* in between the networking substrate and the terminal application domain [41]. Regarding architectures that focus on CM or SHM domains, we highlight Lamonaca et al. [42], which defines application, event detection, signal processing, and sensing as the layers of the SHM Systems. Its goal is to create a framework where all IoT systems for SHM can fit.

Finally, five-layer IoT architectures introduce an additional level above the application services known as *Business layer*, since it orchestrates the entire IoT system as a whole [18]. In the scope of CM, there is a known five-steps architectural guideline for implementing IoT-based systems for SHM. The steps – or layers – are sensing, gateway, network, control, and graphic interface [6], [43].

Diverging from the discussed approaches, other researchers deployed CM systems that do not fit in any of the mentioned layer-based IoT architectures. Wang et al. [4] proposes and implements an architecture to perform continuous CM. It comprises three tiers: edge, platform, and enterprise. The edge tier, via the IIoT gateway, connects to the cloud and platform tier, and the enterprise tier accesses data from the platform tier. Qian et al. [44] presents an IoT-based approach to condition monitoring of the wave power generation system, though, they do not present a software architecture to support it. Yang et al. [45] proposes a cloud-based CM platform for industrial applications, their system also encompasses edge computing nodes for performing tasks that require low latency.

The proposed architecture differs from the ones in literature mainly due to its toolchain aspect. Each layer of the presented architecture in Section III describes a high-level functionality that is needed to implement a system feature. In turn, system features are enabled by toolchains. What makes the system design and its features clear and objective also facilitates integrating more features – *i.e.*, toolchains – into the system and managing the ones already in place due to its modularity and loose-coupling features. Additionally, we do not include infrastructural components – such as edge, cloud, gateway, and network – as architectural layers. The layers should reflect the function it enables, not the physical location of that layer's computational tasks. Finally, as far as we know, no system or architecture in literature encompasses all the features that are included in our architecture, those are: Management of applications, management of devices, and management of energy.

VIII. CONCLUSION

In this paper, we proposed an architectural approach for CM scenarios based on software toolchains that are responsible for a defined set of tasks. Different toolchains support the engineering process of the SoS and enable the separation of concerns for the different actors that typically interact with complex and scalable CM scenarios. Following the guidelines of the Arrowhead Tools project, we first established a number of goals that our architecture aims to accomplish and we then developed several engineering tools that compose the proposed toolchains at all layers of the architecture. We then experimented with the four developed toolchains in a real SHM scenario that involves inertial and gas sensors installed over a bridge. The tests proved the efficacy of the implemented toolchains, in particular the improved automation of the whole engineering process for wider fruition by different stakeholders at different levels of abstraction.

Future Directions: The primary purpose of toolchains is to provide a secure and efficient way of designing and deploying software and other applications. The various solutions proposed in this paper are designed to address specific problems involving different types of users. Although our approach is reasonable, there is still ample room for improvement by empowering the user to create new toolchains that meet his specific requirements in the most relevant way possible. It is imperative to understand that not all users are developers capable of working at the code level to modify and create their systems. For this reason, it might be interesting to move the focus to a no-code or low-code platform that enables the composition of new toolchains based on a set of existing components. This could involve developing new visual interfaces or improving automation and integration capabilities. One of the biggest challenges will be achieving an optimal balance between the abstraction offered to users and the actual performance of the generated solution. This trade-off becomes even more significant in the specific case of toolchains for CM since, in many cases, we have strong scalability and processing power requirements. In addition to usability, another aspect to be considered critical in the management of toolchains is security. Although this aspect has already been covered in our work, it should be emphasized that maintaining a vulnerability-free system is not easy to achieve, and each solution has its own drawbacks. One of the most complex issues is ensuring the integrity of the toolchain despite its security being linked to the security of its components. To address these issues, it is crucial to design a platform capable of verifying the quality and the provenance of the single components and also performing regular software updates to be sure that the code is patched against the newest exploits.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "Understanding the Internet of Things: Definition," potentials, and societal role of a fast evolving paradigm," *Ad Hoc Netw.*, vol. 56, pp. 122–140, Apr. 2017.

[2] R. B. Randall, *Vibration-Based Condition Monitoring: Industrial, Automotive and Aerospace Applications*. Hoboken, NJ, USA: Wiley, 2021.

[3] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review," *Renew. Energy*, vol. 133, pp. 620–635, Apr. 2019.

[4] G. Wang, M. Nixon, and M. Boudreaux, "Toward cloud-assisted industrial IoT platform for large-scale continuous condition monitoring," *Proc. IEEE*, vol. 107, no. 6, pp. 1193–1205, Jun. 2019.

[5] A. Pegoretti, *Structural Health Monitoring: Current State and Future Trends*. Warrendale, PA, USA: SAE, 2018.

[6] C. Scuro, P. F. Sciammarella, F. Lamonaca, R. S. Olivito, and D. L. Carni, "IoT for structural health monitoring," *IEEE Instrum. Meas. Mag.*, vol. 21, no. 6, pp. 4–14, Dec. 2018.

[7] A. Massaro, "Information technology infrastructures supporting industry 5.0 facilities," in *Electronics in Advanced Research Industries: Industry 4.0 to Industry 5.0 Advances*. IEEE, 2022, pp. 51–101.

[8] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Toumura, and K. Kajimoto. (Apr. 2020). *Web of Things (WOT) Architecture*. W3C Recommendation. [Online]. Available: https://www.w3.org/TR/wot-architecture/

[9] L. Sciullo, L. Gigli, F. Montori, A. Trotta, and M. D. Felice, "A survey on the web of things," *IEEE Access*, vol. 10, pp. 47570–47596, 2022.

[10] C. Aguzzi, L. Gigli, L. Sciullo, A. Trotta, F. Zonzini, L. De Marchi, M. Di Felice, A. Marzani, and T. S. Cinotti, "MODRON: A scalable and interoperable web of things platform for structural health monitoring," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–7.

[11] G. Urgese, P. Azzoni, J. V. Deventer, J. Delsing, and E. Macii, "An engineering process model for managing a digitalised life-cycle of products in the industry 4.0," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp. (NOMS)*, Apr. 2020, pp. 1–6.

[12] G. Kulcsár, P. Varga, M. S. Tatara, F. Montori, M. A. Inigo, G. Urgese, and P. Azzoni, "Modeling an industrial revolution: How to manage large-scale, complex IoT ecosystems?" in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2021, pp. 896–901.

[13] ISO Central Secretary, *Industrial Systems, Installations and Equipment and Industrial Products—Structuring Principles and Reference Designations*,'' Standard IEC 81346, Int. Org. Standardization, Geneva, CH, USA, 2019.

[14] G. Kulcsár, M. S. Tatara, and F. Montori, ''Toolchain modeling: Comprehensive engineering plans for industry 4.0,'' in *Proc. IECON 46th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2020, pp. 4541–4546.

[15] D. G. S. Pivoto, L. F. F. De Almeida, R. Da Rosa Righi, J. J. P. C. Rodrigues, A. B. Lugli, and A. M. Alberti, ''Cyber-physical systems architectures for industrial Internet of Things applications in industry 4.0: A literature review,'' *J. Manuf. Syst.*, vol. 58, pp. 176–192, Jan. 2021.

[16] J. Delsing, *IoT Automation: Arrowhead Framework*. Boca Raton, FL, USA: CRC Press, 2017.

[17] G. Kulcsár, K. Koltai, S. Tanyi, B. Péceli, A. Horváth, Z. Micskei, and P. Varga, ''From models to management and back: Towards a system-of-systems engineering toolchain,'' in *Proc. IEEE/IFIP Netw. Operations Manage. Symp. (NOMS)*, Apr. 2020, pp. 1–6.

[18] I. Zyrianoff, A. Heideker, D. Silva, J. Kleinschmidt, J.-P. Soininen, T. S. Cinotti, and C. Kamienski, ''Architecting and deploying IoT smart applications: A performance–oriented approach,'' *Sensors*, vol. 20, no. 1, p. 84, Dec. 2019.

[19] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, ''Big IoT data analytics: Architecture,'' opportunities, and open research challenges,'' *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[20] N. Testoni, C. Aguzzi, V. Arditi, F. Zonzini, L. De Marchi, A. Marzani, and T. S. Cinotti, ''A sensor network with embedded data processing and data-to-cloud capabilities for vibration-based real-time SHM,'' *J. Sensors*, vol. 2018, pp. 1–12, Jul. 2018.

[21] F. Zonzini, L. De Marchi, and N. Testoni, ''A small footprint, low power, and light weight sensor node and dedicated processing for modal analysis,'' in *Convegno Nazionale Sensori*. London, U.K.: Springer, 2018, pp. 361–370.

[22] C. Bruno, A. Licciardello, G. A. M. Nastasi, F. Passaniti, C. Brigante, F. Sudano, A. Faulisi, and E. Alessi, ''Embedded artificial intelligence approach for gas recognition in smart agriculture applications using low cost MOX gas sensors,'' in *Proc. Smart Syst. Integr. (SSI)*, Apr. 2021, pp. 1–5.

[23] M. Noura, M. Atiquzzaman, and M. Gaedke, ''Interoperability in Internet of Things: Taxonomies and open challenges,'' *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 796–809, 2019.

[24] F. Montori, I. Zyrianoff, L. Gigli, R. Venanzi, S. Sindaco, C. Aguzzi, F. Zonzini, M. Zauli, N. Testoni, E. Alessi, M. D. Felice, L. Bononi, P. Bellavista, L. De Marchi, and T. S. Cinotti, ''A toolchain architecture for condition monitoring using the eclipse arrowhead framework,'' in *Proc. 47th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2021, pp. 1–6.

[25] R. Venanzi, F. Montori, P. Bellavista, and L. Foschini, ''Industry 4.0 solutions for interoperability: A use case about tools and tool chains in the arrowhead tools project,'' in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Sep. 2020, pp. 429–433.

[26] I. Zyrianoff, L. Gigli, F. Montori, C. Kamienski, and M. D. Felice, ''Two-way integration of service-oriented systems-of-systems with the web of things,'' in *Proc. 47th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2021, pp. 1–6.

[27] I. Zyrianoff, L. Gigli, F. Montori, C. Aguzzi, S. Kaebisch, and M. Di Felice, ''Seamless integration of RESTful web services with the web of things,'' in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Affiliated Events (PerCom Workshops)*, Mar. 2022, pp. 427–432.

[28] K. Georgiou, S. Xavier-De-Souza, and K. Eder, ''The IoT energy challenge: A software perspective,'' *IEEE Embedded Syst. Lett.*, vol. 10, no. 3, pp. 53–56, Sep. 2018.

[29] I. Zyrianoff, A. Trotta, L. Sciullo, F. Montori, and M. Di Felice, ''IoT edge caching: Taxonomy,'' use cases and perspectives,'' *IEEE Internet Things Mag.*, vol. 5, no. 3, pp. 12–18, Sep. 2022.

[30] J. Kristan, P. Azzoni, L. Römer, S. E. Jeroschewski, and E. Londero, ''Evolving the ecosystem: Eclipse arrowhead integrates eclipse IoT,'' in *Proc. IEEE/IFIP Netw. Operations Manage. Symp. (NOMS)*, Apr. 2022, pp. 1–6.

[31] M. Tarozzi, G. Pignagnoli, and A. Benedetti, ''Identification of damage-induced frequency decay on a large-scale model bridge,'' *Eng. Struct.*, vol. 221, 2020, Art. no. 111039.

[32] C. A. Tokognon, B. Gao, G. Y. Tian, and Y. Yan, ''Structural health monitoring framework based on Internet of Things: A survey,'' *IEEE Internet Things J.*, vol. 4, no. 3, pp. 619–635, Feb. 2017.

[33] M. Azimi, A. D. Eslamlou, and G. Pekcan, ''Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review,'' *Sensors*, vol. 20, vol. 10, p. 2778, 2020.

[34] J. P. Lynch, C. R. Farrar, and J. E. Michaels, ''Structural health monitoring: Technological advances to practical implementations,'' *Proc. IEEE*, vol. 104, no. 8, pp. 1508–1512, Aug. 2016.

[35] A. Verma, S. Prakash, V. Srivastava, A. Kumar, and S. C. Mukhopadhyay, ''Sensing, controlling, and IoT infrastructure in smart building: A review,'' *IEEE Sensors J.*, vol. 19, no. 20, pp. 9036–9046, Jun. 2019.

[36] Z. Chen, X. Zhou, X. Wang, L. Dong, and Y. Qian, ''Deployment of a smart structural health monitoring system for long-span arch bridges: A review and a case study,'' *Sensors*, vol. 17, no. 9, p. 2151, 2017.

[37] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, ''IoT architecture challenges and issues: Lack of standardization,'' in *Proc. Future Technol. Conf. (FTC)*, Dec. 2016, pp. 731–738.

[38] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, ''Research on the architecture of Internet of Things,'' in *Proc. 3rd Int. Conf. Adv. Comput. Theory Eng. (ICACTE)*, vol. 5, 2010, pp. V5-484–V5-487.

[39] O. Said and M. Masud, ''Towards Internet of Things: Survey and future vision,'' *Int. J. Comput. Netw.*, vol. 5, no. 1, pp. 1–17, 2013.

[40] F. Zonzini, C. Aguzzi, L. Gigli, L. Sciullo, N. Testoni, L. De Marchi, M. Di Felice, T. S. Cinotti, C. Mennuti, and A. Marzani, ''Structural health monitoring and prognostic of industrial plants and civil structures: A sensor to cloud architecture,'' *IEEE Instrum. Meas. Mag.*, vol. 23, no. 9, pp. 21–27, Dec. 2020.

[41] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, ''A survey on Internet of Things: Architecture,'' enabling technologies, security and privacy, and applications,'' *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[42] F. Lamonaca, C. Scuro, D. Grimaldi, R. S. Olivito, P. F. Sciammarella, and D. L. Carnì, ''A layered IoT-based architecture for a distributed structural health monitoring system system,'' *Acta Imeko*, vol. 8, no. 2, pp. 45–52, 2019.

[43] C. Scuro, F. Lamonaca, S. Porzio, G. Milani, and R. S. Olivito, ''Internet of Things (IoT) for masonry Structural Health Monitoring (SHM): Overview and examples of innovative systems,'' *Construct. Building Mater.*, vol. 290, Jul. 2021, Art. no. 123092.

[44] P. Qian, B. Feng, D. Zhang, X. Tian, and Y. Si, ''IoT-based approach to condition monitoring of the wave power generation system,'' *IET Renew. Power Gener.*, vol. 13, no. 12, pp. 2207–2214, Sep. 2019.

[45] H. Yang, Z. Sun, G. Jiang, F. Zhao, X. Lu, and X. Mei, ''Cloud-manufacturing-based condition monitoring platform with 5G and standard information model,'' *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6940–6948, Apr. 2021.

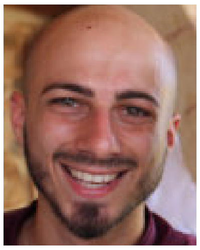
FEDERICO MONTORI (Member, IEEE), received the B.S. and M.S. degrees (summa cum laude), in computer science, and the Ph.D. degree in computer science and engineering from the University of Bologna, Italy, in 2012, 2015, and 2019, respectively. Since 2022, he is a Junior Assistant Professor with the University of Bologna. He has been a Visiting Researcher at the Swinburne University of Technology, Australia and Luleå Tekniska Universitet, Sweden. He participated in several EU projects and he is now a WP Leader for the H2020 Project Arrowhead Tools. His primary research interests include mobile crowdsensing (MCS), pervasive and mobile computing, the IoT automation, and data analytics.

IVAN ZYRIANOFF received the B.S. degree in computer science and the M.S. degree in information engineering from the Federal University of ABC, Brazil. He is currently pursuing the Ph.D. degree with the University of Bologna. He is a member of the IoT Prism Laboratory. He was involved in the SWAMP Project, an EU-Brazil collaborative research project that developed the IoT-based methods and approaches for smart water management in precision irrigation. Currently, he is an Active Member of the Arrowhead Tools project. His current research topics encompass interoperability for the Internet of Things, edge computing, and the IoT Architectures.

LORENZO GIGLI (Graduate Student Member, IEEE) received the master's degree (summa cum laude) in computer science from the University of Bologna, Italy, in 2019. He worked as a Research Fellow on the MAC4PRO Project (INAIL) at the Department of Computer Science and Engineering (DISI), University of Bologna. Currently, he is enrolled in a Ph.D. Program in engineering and information technology at the University of Bologna. He is a part of the IoT PRISM Laboratory directed by Prof. Marco Di Felice. His research interests include interoperability for the Internet of Things, the IoT and cloud architectures, and distributed systems.

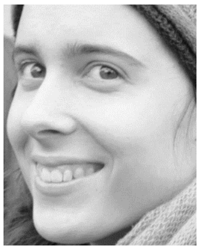
ALESSANDRO CALVIO received the B.S. and M.S. degrees (cum laude) in computer engineering from the University of Bologna, Italy, in 2018 and 2021, respectively. He is currently a Research Fellow with the University of Bologna, where he is involved in several EU projects comprises the H2020 Arrowhead Tools Project. His main research interests include the IoT, data analysis, and big-data architectures for smart city contexts.

RICCARDO VENANZI (Member, IEEE) received the Ph.D. degree in computer engineering from the University of Ferrara, Italy, in 2019. He is currently a Postdoctoral Research Fellow with DISI, Department of Computer Science and Engineering, University of Bologna, Italy. His research interests include span cloud and edge computing, the Internet of Things, distributed systems, mobile computing, and middleware, and the Industrial Internet of Things and Industry 4.0.



SIMONE SINDACO received the B.S. and M.S. degrees (summa cum laude) in electronic engineering. Since September 2019, he has been a Research Fellow with the University of Bologna, Italy. He participated in several EU-H2020 projects such as SWAMP and Arrowhead Tools. His research interests include the design of sensors network for structural and environmental health monitoring purposes and the design of embedded devices for the railway sector (ERTMS) in partnership with the railway Italian company (RFI).

LUCA SCIULLO (Member, IEEE) received the master's degree (summa cum laude) in computer science and the Ph.D. degree in computer science and engineering from the University of Bologna, Italy, in 2017 and 2021, respectively. He is a Postdoctoral Researcher with the University of Bologna. He was a Visiting Researcher at the Huawei European Research Center of Munich, Germany. He is a part of the IoT Prism Laboratory directed by Prof. Marco Di Felice and Prof. Luciano Bononi. His research interests include wireless systems and protocols for emergency scenarios, wireless sensor networks, the IoT systems, and the Web of Things.



FEDERICA ZONZINI (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering, and the Ph.D. degree in Structural and Environmental Health Monitoring and Management (SEHM2) from the University of Bologna, in 2016, 2018, and 2022, respectively. Her main research interests include advanced signal processing techniques for structural health monitoring applications, encompassing graph signal processing, compressive sensing, and artificial intelligence.

MATTEO ZAULI (Member, IEEE) received the B.S. and the M.S. degrees in electronics engineering from the University of Bologna, Italy, where he is currently pursuing the Ph.D. degree in engineering and information technology for Structural and Environmental Monitoring and Risk Management (EIT4SEMM). His research interests focus on signal processing, compressed sensing, neural networks, embedded systems development, and ultrasonic measurements.

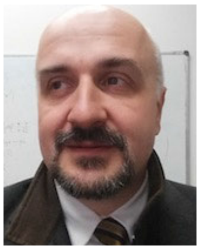
NICOLA TESTONI (Member, IEEE) received the M.Sc. degree in microelectronics and the Ph.D. degree in information technology from Bologna University, in 2004 and 2008, respectively. He is currently an Adjunct Professor with the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi", Bologna University. His research interests include guided waves, analog circuit design, non-linear signal processing, wavelet theory and applications, neural signal denoising, and event sorting.

ALESSANDRO BERTACCHINI (Member, IEEE) received the master's degree in computer science and the Ph.D. degree in industrial management engineering from the University of Modena and Reggio Emilia, Italy, in 2001 and 2005, respectively, where he is currently an Assistant Professor His research interests are focused on the design of embedded systems for industrial and safety-critical applications and the development of low-power energy harvesting systems. He authored and coauthored several peer-reviewed technical papers on these topics. He is a member of the IEEE-IES (2009) and IEEE IAS (2014).

ELISA LONDERO received the M.Sc. degree in physics from Trieste University, Italy, in 2008, and the Ph.D. degree in condensed matter physics from the Chalmers University of Technology, Sweden, in 2012. From 2013 to 2016, she was a Postdoctoral Research Associate with the Wigner Research Centre for Physics and Optics, Budapest, Hungary. From 2016 to 2020, she was a Research Fellow at Trieste Observatory, Italy. Since 2020, she has been working with the Eurotech, Research and Development Department, Amaro, Italy, as an Applied Research Scientist.

ENRICO ALESSI received the M.D. degree in electronics engineering and the M.D. degree in computer science. He is currently a Research and Development Application Manager and a Senior Member of technical staff at the STMicroelectronics with 20 years of experience in large companies working mainly in the field of ICT and microelectronics. He has matured experience in sensors, biosensors, and medical systems for IVD. He was a First author and a Technical Coordinator of several national and European funding research programs in the domain of environmental sensors, human body sensors, and biosensors for DNA Analysis such as Lab-on-chips for DNA Amplification and detection and Lab-on-disk for sample preparation.

MARCO DI FELICE (Member, IEEE) received the Laurea (summa cum laude) and Ph.D. degrees in computer science from the University of Bologna, in 2004 and 2008, respectively. He is a Full Professor of computer science with the University of Bologna, where he is the Co-Director of the IoT PRISM Laboratory. He was a Visiting Researcher with the Georgia Institute of Technology, Atlanta, GA, USA, and with Northeastern University, Boston, MA, USA. He authored more than 120 articles on wireless and mobile systems. His research interests include self-organizing wireless networks, unmanned aerial systems, the IoT, WoT, and context-aware computing. He achieves three best paper awards for his scientific production. He is the Associate Editor of the IEEE INTERNET OF THINGS JOURNAL.



LUCIANO BONONI received the M.Sc (summa cum laude) degree, in 1997 and the Ph.D. degree, in 2001. He was the Founder of the Wireless and Mobile Applications Laboratory, in 2010. He is a Full Professor with the Department of Computer Science and Engineering, University of Bologna. He is actively involved in national and international research projects and collaborations with international Universities and Industry partners. His research fields include wireless systems and networks, the Internet of Things, smart mobility and mobile applications, simulation, and digital twins. He is a member of the Doctoral Board for the Ph.D. Program in computer science and engineering at the University of Bologna. From 2012 to 2015, he was elected member of the academic senate of the University of Bologna.

PAOLO BELLAVISTA (Senior Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2001, where he is currently a Full Professor His research interests include middleware for mobile computing, QoS management in the cloud continuum, infrastructures for big data processing in industrial environments, and performance optimization in wide-scale and latency-sensitive deployment environments. He served as the Scientific Coordinator for the H2020 IoTwins Project. He serves on the Editorial Boards for IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON SERVICES COMPUTING, *ACM CSUR*, *ACM TIOT*, and *PMC* (Elsevier).

LUCA DE MARCHI (Senior Member, IEEE) is currently an Associate Professor of electronics with the University of Bologna, Bologna, Italy. He is the Coordinator with the University of Bologna Ph.D. Program in engineering and information technology for Structural and Environmental Monitoring and Risk Management-EIT4SEMM. His current research interests include intelligent sensor systems and embedded signal processing, with a particular emphasis on vibration and ultrasound sensing. In this field, he has published more than 180 articles in international journals or in proceedings of international conferences and he holds two patents.

ALESSANDRO MARZANI received the M.Sc. (Laurea) degree in civil engineering from the University of Bologna, Italy, in 2001, the M.Sc. degree in structural engineering from the University of California San Diego, USA, in 2004, and the Ph.D. degree in engineering of materials and structures from the University of Calabria, Italy, in 2005. He is currently a Professor of structural mechanics with the Department of Civil, Chemical, Environmental and Material Engineering, University of Bologna. His research interests include non-destructive evaluation techniques of materials and structures, structural monitoring, linear and nonlinear ultrasonic guided wave propagation, structural optimization and identification strategies, and structured materials for wave propagation control metamaterials.

PAOLO AZZONI received the master's degree in computer science and the second master's degree in intelligent systems. He is the Secretary-General at Inside-IA (formerly Artemis-IA), the industry association that serves as the European Technology Platform for research, design, and innovation on Intelligent Digital Systems and their technology ecosystems. In this context, he is the Chairperson of the ECS Strategic Research and Innovation Agenda, a funding-agnostic document describing the major challenges and priorities in the ECS domain for the next 10 years. He is also the Head of European Technology Programs at EUROTECH Group, planning and directing industrial research projects, investigating technologies beyond the state of the art in the domains of cyber-physical systems, intelligent systems, machine-to-machine technologies, edge computing, and the Internet of Things and digitalization solutions.

TULLIO SALMON CINOTTI received the degree in electrical engineering from the University of Bologna, in 1974. He is a Honorary Professor with the School of Engineering and Architecture, University of Bologna, currently teaching logic design. He was the Director of ARCES, an Inter-Department Research Center at the University of Bologna, from 2015 to 2018. He is the coauthor of researchers from Intel Labs, Nokia Research, Siemens Corporate Technology, STMicroelectronics, Telecom Italia Lab, VTT, Politecnico di Milano, University of Kent, and University of Westminster. His research interests include digital systems and go from embedded systems to smart spaces and semantics-based data distribution architectures for cyber-physical systems. He is the Coordinator at the University of Bologna's participation in European research initiatives in the areas of open cultural heritage, electric mobility, smart agriculture, smart environments, and SoS engineering.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement